KEYFACTOR

2023 | Keyfactor

Keyfactor Command 11.1

Documentation Suite

Table of Contents

1.0 Introduction **1**

2.0 Reference Guide **2**

2.1 Using the Management Portal 2
2.1.1 Authentication and Authorization 5
2.1.2 Dashboard 6
2.1.2.1 Dashboard: CA Status 11
2.1.2.2 Dashboard: Collections 12
2.1.2.3 Dashboard: Certificates by Signing Algorithm 13
2.1.2.4 Dashboard: Number of SSH Keys per Type 14
2.1.2.5 Dashboard: Recent Certificate Store Jobs 15
2.1.2.6 Dashboard: Revocation Monitoring 16
2.1.2.7 Dashboard: SSL Endpoints 17
2.1.2.8 Dashboard: SSL Orchestrator Job Status 18
2.1.3 Certificate Search and Collections 19
2.1.3.1 Certificate Details 19
2.1.3.2 Certificate Search Page 34
2.1.3.3 Certificate Operations 45
2.1.3.4 Add Certificate 74
2.1.3.5 Certificate Collection Manager 85
2.1.4 Reports 91
2.1.4.1 Certificate Count by Template 94
2.1.4.2 Certificate Count by User per Template 95
2.1.4.3 Certificate Count Grouped by Single Metadata Field 97
2.1.4.4 Certificate Issuance Trends with Metadata 98
2.1.4.5 Certificates by Key Strength 100
2.1.4.6 Certificates by Revoker 101
2.1.4.7 Certificates by Type and Java Keystore 102
2.1.4.8 Certificates Found at TLS/SSL Endpoints 103
2.1.4.9 Certificates in Collection 104
2.1.4.10 Expiration Report 105
2.1.4.11 Expiration Report by Days 108
2.1.4.12 Full Certificate Extract Report 110
2.1.4.13 Issued Certificates per Certificate Authority 112
2.1.4.14 Monthly Executive Report 113
2.1.4.15 PKI Status for Collection 114
2.1.4.16 Revoked Certificates in Certificate Stores 120
2.1.4.17 SSH Key Usage 121
2.1.4.18 SSH Keys by Age 122
2.1.4.19 SSH Keys with Root Logon Access 123
2.1.4.20 SSH Trusted Public Keys with No Known Private Keys 124
2.1.4.21 Statistical Report 125
2.1.4.22 Report Manager 126
2.1.5 Enrollment 135
2.1.5.1 CSR Enrollment 136
2.1.5.2 CSR Generation 142
2.1.5.3 Pending CSRs 146
2.1.5.4 PFX Enrollment 146
2.1.5.5 Certificate Requests 163
2.1.6 Alerts 166
2.1.6.1 Expiration Alerts 167
2.1.6.2 Pending Certificate Request Alerts 178
2.1.6.3 Issued Certificate Request Alerts 188
2.1.6.4 Denied Certificate Request Alerts 197
2.1.6.5 Key Rotation Alerts 203
2.1.6.6 Revocation Monitoring 210
2.1.6.7 Using Event Handlers 218

2.1.7 Workflow ... 229
2.1.7.1 Workflow Definitions ... 230
2.1.7.2 Workflow Instances ... 305
2.1.7.3 My Workflows ... 324
2.1.8 Locations ... 348
2.1.8.1 Certificate Authorities ... 349
2.1.8.2 Certificate Templates ... 379
2.1.8.3 Certificate Stores ... 408
2.1.8.4 SSL Discovery ... 453
2.1.9 Orchestrators ... 481
2.1.9.1 Orchestrator Auto-Registration ... 485
2.1.9.2 Orchestrator Management ... 496
2.1.9.3 Orchestrator Job Status ... 510
2.1.9.4 Orchestrator Blueprints ... 521
2.1.9.5 Mac Auto-Enrollment ... 524
2.1.10 SSH ... 525
2.1.10.1 My SSH Key ... 531
2.1.10.2 Service Account Keys ... 542
2.1.10.3 Unmanaged SSH Keys ... 556
2.1.10.4 Server Manager ... 560
2.1.10.5 SSH Permissions ... 597
2.1.11 System Settings ... 600
2.1.11.1 Application Settings ... 601
2.1.11.2 Security Roles and Claims ... 622
2.1.11.3 Certificate Store Types ... 700
2.1.11.4 Certificate Metadata ... 710
2.1.11.5 Audit Log ... 716
2.1.11.6 Event Handler Registration ... 739
2.1.11.7 Privileged Access Management (PAM) ... 742
2.1.11.8 Identity Providers ... 754
2.1.11.9 SMTP Configuration ... 765
2.1.11.10 Component Installations ... 767
2.1.11.11 Licensing ... 768
2.2 Operations ... 770
2.2.1 Extending and Customizing Keyfactor Command ... 770
2.2.1.1 Editing Job Completion Handlers ... 771
2.2.1.2 Customize the Management Portal Banner Logo ... 772
2.2.1.3 Keyfactor Command Appsetting.json Files ... 773
2.2.1.4 Keyfactor Command Service Executable ... 788
2.2.1.5 Configuring Key Recovery for Keyfactor Command ... 789
2.2.2 SSH Reference ... 790
2.2.2.1 SSH-Bash Orchestrator Job History Warning Resolution ... 790
2.2.2.2 SSH-SSSD Case Sensitivity Flag ... 791
2.2.3 System Alerts ... 793
2.2.4 Log Monitoring ... 794
2.2.4.1 Editing NLog ... 796
2.2.4.2 Audit Log Output to a Centralized Logging Solution ... 805
2.2.4.3 Keyfactor Command Windows Event IDs ... 806
2.2.5 License Expiration Monitoring and Rotation ... 817
2.2.6 Disaster Recovery ... 819
2.2.6.1 SQL Encryption Key Backup ... 821
2.2.7 SQL Database Migration ... 823
2.2.8 Troubleshooting ... 824
2.2.8.1 Disable Loopback Checking ... 832
2.3 Appendices ... 834
2.3.1 Appendix - References ... 834
2.3.2 Appendix - Third-Party Notices for Keyfactor Command Software ... 834
2.3.2.1 Apache 2.0 License Text: ... 836
2.3.2.2 BSD License Text: ... 839
2.3.2.3 MIT License Text: ... 839
2.3.2.4 Microsoft Public License (MS-PL) Text: ... 840
2.3.2.5 Microsoft Reciprocal License (MS-RL) Text: ... 841

3.0 Keyfactor API Reference 843
3.1 Overview 843
3.2 Authenticating to the Keyfactor API 844
3.3 Transaction Security 849
3.4 Endpoint Common Features 849
3.5 Versioning 852
3.6 Keyfactor API Endpoints 852
3.6.1 Agents 853
3.6.1.1 GET Agents ID 854
3.6.1.2 GET Agents 858
3.6.1.3 POST Agents Reset 864
3.6.1.4 POST Agents Approve 864
3.6.1.5 POST Agents Disapprove 865
3.6.1.6 POST Agents ID Reset 866
3.6.1.7 POST Agents ID FetchLogs 866
3.6.1.8 POST Agents Set Auth Certificate Reenrollment 867
3.6.2 Agent BluePrint 869
3.6.2.1 DELETE Agent BluePrint ID 870
3.6.2.2 GET Agent BluePrint ID 871
3.6.2.3 GET Agent BluePrint 872
3.6.2.4 GET Agent BluePrint ID Jobs 873
3.6.2.5 GET Agent BluePrint ID Stores 878
3.6.2.6 POST AgentBluePrint ApplyBluePrint 881
3.6.2.7 POST AgentBluePrint GenerateBluePrint 882
3.6.3 Agent Pools 883
3.6.3.1 DELETE Agent Pools ID 883
3.6.3.2 GET Agent Pools ID 884
3.6.3.3 GET Agent Pools 886
3.6.3.4 POST Agent Pools 889
3.6.3.5 PUT Agent Pools 891
3.6.3.6 GET Agent Pools Agents 894
3.6.4 Alerts 896
3.6.4.1 Alerts Denied 896
3.6.4.2 Alerts Expiration 927
3.6.4.3 Alerts Issued 966
3.6.4.4 Alerts Key Rotation 1001
3.6.4.5 Alerts Pending 1034
3.6.5 AppSetting 1075
3.6.5.1 GET AppSetting 1075
3.6.5.2 GET AppSetting ID 1077
3.6.5.3 PUT AppSetting 1079
3.6.5.4 PUT AppSetting ID Set 1081
3.6.5.5 PUT AppSetting Name Set 1083
3.6.6 Audit 1085
3.6.6.1 GET Audit ID 1085
3.6.6.2 GET Audit ID Validate 1090
3.6.6.3 GET Audit 1091
3.6.6.4 GET Audit Download 1097
3.6.6.5 GET Audit Related Entities 1101
3.6.7 Certificates 1107
3.6.7.1 GET Certificates ID Security 1109
3.6.7.2 GET Certificates ID Validate 1111
3.6.7.3 GET Certificates Locations ID 1116
3.6.7.4 GET Certificates Identity Audit ID 1119
3.6.7.5 DELETE Certificates ID 1123
3.6.7.6 GET Certificates ID 1124
3.6.7.7 GET Certificates Metadata Compare 1136
3.6.7.8 GET Certificates ID History 1138
3.6.7.9 DELETE Certificates 1140
3.6.7.10 GET Certificates 1141
3.6.7.11 PUT Certificates Metadata 1156

3.6.7.12 PUT Certificates Metadata All ... 1158
3.6.7.13 POST Certificates Import ... 1162
3.6.7.14 POST Certificates Revoke ... 1166
3.6.7.15 POST Certificates Analyze ... 1168
3.6.7.16 POST Certificates Recover ... 1169
3.6.7.17 POST Certificates Download ... 1173
3.6.7.18 POST Certificates Revoke All ... 1176
3.6.7.19 DELETE Certificates Query ... 1179
3.6.7.20 DELETE Certificates Private Key ... 1180
3.6.7.21 DELETE Certificates Private Key ID ... 1181
3.6.8 Certificate Authority ... 1182
3.6.8.1 DELETE Certificate Authority ID ... 1185
3.6.8.2 GET Certificate Authority ID ... 1186
3.6.8.3 GET Certificate Authority ... 1201
3.6.8.4 POST Certificate Authority ... 1216
3.6.8.5 PUT Certificate Authority ... 1248
3.6.8.6 POST Certificate Authority Test ... 1280
3.6.8.7 POST Certificate Authority PublishCRL ... 1284
3.6.8.8 GET Certificate Authority Source Count ... 1284
3.6.8.9 GET Certificate Authority Available Forests ... 1285
3.6.8.10 GET Certificate Authority Health Monitoring Schedule ... 1286
3.6.8.11 GET Certificate Authority Alert Recipients CA Health Recipients ... 1286
3.6.8.12 POST Certificate Authority Alert Recipients CA Health Recipients ... 1287
3.6.8.13 GET Certificate Authority Alert Recipients CA Health Recipients ID ... 1288
3.6.8.14 DELETE Certificate Authority Alert Recipients CA Health Recipients ID ... 1289
3.6.8.15 PUT Certificate Authority Alert Recipients CA Health Recipients ID ... 1290
3.6.8.16 DELETE Certificate Authority Alert Recipients CA Threshold Recipients ID ... 1290
3.6.8.17 GET Certificate Authority Alert Recipients CA Threshold Recipients ... 1291
3.6.8.18 GET Certificate Authority Alert Recipients CA Threshold Recipients ID ... 1292
3.6.8.19 POST Certificate Authority Alert Recipients CA Threshold Recipients ... 1292
3.6.8.20 PUT Certificate Authority Alert Recipients CA Threshold Recipients ID ... 1293
3.6.8.21 POST Certificate Authority Import ... 1294
3.6.9 Certificate Collections ... 1295
3.6.9.1 GET Certificate Collections ... 1296
3.6.9.2 POST Certificate Collections ... 1299
3.6.9.3 PUT Certificate Collections ... 1305
3.6.9.4 GET Certificate Collections ID ... 1309
3.6.9.5 DELETE Certificate Collection ID ... 1311
3.6.9.6 GET Certificate Collections Name ... 1311
3.6.9.7 POST Certificate Collections Copy ... 1314
3.6.9.8 GET Certificate Collection Nav Items ... 1320
3.6.9.9 PUT Certificate Collection ID Favorite ... 1320
3.6.9.10 GET Certificate Collections List ... 1321
3.6.10 Certificate Stores ... 1324
3.6.10.1 DELETE Certificate Stores ... 1326
3.6.10.2 GET Certificate Stores ... 1327
3.6.10.3 POST Certificate Stores ... 1339
3.6.10.4 PUT Certificate Stores ... 1367
3.6.10.5 DELETE Certificate Stores ID ... 1397
3.6.10.6 GET Certificate Stores ID ... 1398
3.6.10.7 GET Certificate Stores ID Inventory ... 1416
3.6.10.8 GET Certificate Stores Server ... 1418
3.6.10.9 POST Certificate Stores Server ... 1421
3.6.10.10 PUT Certificate Stores Server ... 1426
3.6.10.11 PUT Certificate Stores Password ... 1431
3.6.10.12 PUT Certificate Stores Discovery Job ... 1434
3.6.10.13 PUT Certificate Stores Assign Container ... 1440
3.6.10.14 POST Certificate Stores Approve ... 1451
3.6.10.15 POST Certificate Stores Schedule ... 1462
3.6.10.16 POST Certificate Stores Reenrollment ... 1465
3.6.10.17 POST Certificate Stores Certificates Add ... 1467
3.6.10.18 POST Certificate Stores Certificates Remove ... 1473

3.6.11 Certificate Store Containers ... 1476
3.6.11.1 GET Certificate Store Containers ... 1477
3.6.11.2 POST Certificate Store Containers ... 1479
3.6.11.3 PUT Certificate Store Containers ... 1484
3.6.11.4 DELETE Certificate Store Containers ID ... 1509
3.6.11.5 GET Certificate Store Containers ID ... 1509
3.6.12 Certificate Store Types ... 1531
3.6.12.1 DELETE Certificate Store Types ID ... 1531
3.6.12.2 GET Certificate Store Types ID ... 1532
3.6.12.3 GET CertificateStoreTypes Name Name ... 1538
3.6.12.4 DELETE Certificate Store Types ... 1545
3.6.12.5 GET Certificate Store Types ... 1546
3.6.12.6 POST Certificate Store Types ... 1552
3.6.12.7 PUT Certificate Store Types ... 1566
3.6.13 Component Installation ... 1582
3.6.13.1 DELETE Component Installation ID ... 1583
3.6.13.2 GET Component Installation ... 1583
3.6.14 CSR Generation ... 1585
3.6.14.1 DELETE CSR Generation Pending ID ... 1586
3.6.14.2 GET CSR Generation Pending ID ... 1587
3.6.14.3 DELETE CSR Generation Pending ... 1587
3.6.14.4 GET CSR Generation Pending ... 1588
3.6.14.5 POST CSR Generation Generate ... 1589
3.6.15 Custom Job Types ... 1594
3.6.15.1 DELETE Custom Job Types ID ... 1595
3.6.15.2 GET Custom Job Types ID ... 1596
3.6.15.3 GET Custom Job Types ... 1598
3.6.15.4 POST Custom Job Types ... 1600
3.6.15.5 PUT Custom Job Types ... 1604
3.6.16 Enrollment ... 1608
3.6.16.1 GET Enrollment Settings ID ... 1609
3.6.16.2 GET Enrollment CSR Content My ... 1616
3.6.16.3 GET Enrollment PFX Content My ... 1634
3.6.16.4 GET Enrollment Available Renewal ID ... 1653
3.6.16.5 GET Enrollment Available Renewal Thumbprint ... 1656
3.6.16.6 POST Enrollment CSR ... 1658
3.6.16.7 POST Enrollment PFX ... 1665
3.6.16.8 POST Enrollment CSR Parse ... 1683
3.6.16.9 POST Enrollment PFX Deploy ... 1685
3.6.16.10 POST Enrollment PFX Replace ... 1691
3.6.16.11 POST Enrollment Renew ... 1694
3.6.17 Event Handler Registration ... 1697
3.6.17.1 GET Event Handler Registration ... 1698
3.6.17.2 POST Event Handler Registration ... 1700
3.6.17.3 GET Event Handler Registration ID ... 1701
3.6.17.4 PUT Event Handler Registration ID ... 1702
3.6.17.5 DELETE Event Handler Registration ID ... 1704
3.6.18 Extensions Scripts ... 1704
3.6.18.1 DELETE Extensions Scripts ID ... 1705
3.6.18.2 GET Extensions Scripts ID ... 1706
3.6.18.3 GET Extensions Scripts ... 1707
3.6.18.4 POST Extensions Scripts ... 1709
3.6.18.5 PUT Extensions Scripts ... 1713
3.6.19 Identity Providers ... 1715
3.6.19.1 GET Identity Providers ID ... 1716
3.6.19.2 PUT Identity Providers ID ... 1730
3.6.19.3 GET Identity Providers ... 1760
3.6.19.4 GET Identity Providers Types ... 1775
3.6.20 License ... 1776
3.6.20.1 GET License ... 1777
3.6.21 MacEnrollment ... 1780
3.6.21.1 GET MacEnrollment ... 1780

3.6.21.2 PUT MacEnrollment ... 1781
3.6.22 MetadataFields ... 1783
3.6.22.1 DELETE Metadata Fields ID ... 1784
3.6.22.2 GET Metadata Fields ID ... 1785
3.6.22.3 GET Metadata Fields Name ... 1788
3.6.22.4 GET Metadata Fields ID In Use ... 1792
3.6.22.5 DELETE Metadata Fields ... 1793
3.6.22.6 GET Metadata Fields ... 1794
3.6.22.7 POST Metadata Fields ... 1798
3.6.22.8 PUT Metadata Fields ... 1804
3.6.23 Monitoring ... 1811
3.6.23.1 DELETE Monitoring Revocation ID ... 1812
3.6.23.2 GET Monitoring Revocation ID ... 1813
3.6.23.3 GET Monitoring Revocation ... 1817
3.6.23.4 POST Monitoring Revocation ... 1822
3.6.23.5 PUT Monitoring Revocation ... 1830
3.6.23.6 POST Monitoring Resolve OSCP ... 1838
3.6.23.7 POST Monitoring Revocation Test ... 1839
3.6.23.8 POST Monitoring Revocation Test All ... 1841
3.6.24 Orchestrator Jobs ... 1842
3.6.24.1 GET Orchestrator Jobs Job Status Data ... 1843
3.6.24.2 GET Orchestrator Jobs Job History ... 1844
3.6.24.3 GET Orchestrator Jobs Scheduled Jobs ... 1850
3.6.24.4 POST Orchestrator Jobs Custom ... 1854
3.6.24.5 POST Orchestrator Jobs Reschedule ... 1859
3.6.24.6 POST Orchestrator Jobs Unschedule ... 1861
3.6.24.7 POST Orchestrator Jobs Acknowledge ... 1863
3.6.24.8 POST Orchestrator Jobs Custom Bulk ... 1864
3.6.25 PAM Providers ... 1871
3.6.25.1 DELETE PAM Providers ID ... 1871
3.6.25.2 GET PAM Providers ID ... 1872
3.6.25.3 GET PAM Providers Types ... 1888
3.6.25.4 POST PAM Providers Types ... 1891
3.6.25.5 GET PAM Providers ... 1898
3.6.25.6 POST PAM Providers ... 1916
3.6.25.7 PUT PAM Providers ... 1935
3.6.25.8 GET PAM Providers Types ID ... 1954
3.6.26 Permissions ... 1957
3.6.26.1 GET Permissions ... 1958
3.6.27 Permission Sets ... 1959
3.6.27.1 GET Permission Sets ID ... 1961
3.6.27.2 DELETE Permission Sets ID ... 1962
3.6.27.3 GET Permission Sets ... 1963
3.6.27.4 POST Permission Sets ... 1964
3.6.27.5 PUT Permission Sets ... 1965
3.6.28 Reports ... 1967
3.6.28.1 GET Reports ID ... 1968
3.6.28.2 DELETE Reports Custom ID ... 1976
3.6.28.3 GET Reports Custom ID ... 1977
3.6.28.4 DELETE Reports Schedules ID ... 1978
3.6.28.5 GET Reports Schedules ID ... 1979
3.6.28.6 GET Reports ID Parameters ... 1983
3.6.28.7 PUT Reports ID Parameters ... 1986
3.6.28.8 GET Reports ... 1988
3.6.28.9 PUT Reports ... 1991
3.6.28.10 GET Reports Custom ... 1994
3.6.28.11 POST Reports Custom ... 1996
3.6.28.12 PUT Reports Custom ... 1998
3.6.28.13 GET Reports ID Schedules ... 1999
3.6.28.14 POST Reports ID Schedules ... 2004
3.6.28.15 PUT Reports ID Schedules ... 2014
3.6.29 Scheduling ... 2024

3.6.29.1 POST Scheduling ... 2025
3.6.30 Security ... 2027
3.6.30.1 DELETE Security Identities ID ... 2028
3.6.30.2 GET Security Identities ID ... 2029
3.6.30.3 GET Security Identities Lookup ... 2033
3.6.30.4 GET Security Identities ... 2034
3.6.30.5 POST Security Identities ... 2038
3.6.30.6 GET Security Containers ID Roles ... 2039
3.6.30.7 POST Security Containers ID Roles ... 2040
3.6.30.8 GET Security Audit Collections ID ... 2042
3.6.30.9 GET Security My ... 2045
3.6.31 Security Claims ... 2045
3.6.31.1 GET Security Claims ... 2046
3.6.31.2 POST Security Claims ... 2049
3.6.31.3 PUT Security Claims ... 2053
3.6.31.4 GET Security Claims ID ... 2056
3.6.31.5 DELETE Security Claims ID ... 2058
3.6.31.6 GET Security Claims Roles ... 2058
3.6.32 Security Roles Permissions ... 2061
3.6.32.1 GET Security Roles ID Permissions ... 2063
3.6.32.2 GET Security Roles ID Permissions Global ... 2064
3.6.32.3 POST Security Roles ID Permissions Global ... 2065
3.6.32.4 PUT Security Roles ID Permissions Global ... 2067
3.6.32.5 GET Security Roles ID Permissions Containers ... 2069
3.6.32.6 POST Security Roles ID Permissions Containers ... 2070
3.6.32.7 PUT Security Roles ID Permissions Containers ... 2072
3.6.32.8 GET Security Roles ID Permissions Collections ... 2074
3.6.32.9 POST Security Roles ID Permissions Collections ... 2075
3.6.32.10 PUT Security Roles ID Permissions Collections ... 2077
3.6.32.11 GET Security Roles ID Permissions PAM Providers ... 2079
3.6.32.12 PUT Security Roles ID Permissions PAM Providers ... 2080
3.6.33 Security Roles ... 2081
3.6.33.1 DELETE Security Roles ID ... 2082
3.6.33.2 GET Security Roles ID ... 2082
3.6.33.3 GET Security Roles ... 2088
3.6.33.4 POST Security Roles ... 2093
3.6.33.5 PUT Security Roles ... 2105
3.6.33.6 POST Security Roles ID Copy ... 2117
3.6.33.7 PUT Security Roles ID Identities ... 2120
3.6.33.8 GET Security Roles ID Identities ... 2121
3.6.34 SSH ... 2122
3.6.34.1 SSH Keys ... 2127
3.6.34.2 SSH Logons ... 2143
3.6.34.3 SSH Servers ... 2153
3.6.34.4 SSH Server Groups ... 2184
3.6.34.5 SSH Service Accounts ... 2224
3.6.34.6 SSH Users ... 2269
3.6.35 SMTP ... 2294
3.6.35.1 GET SMTP ... 2294
3.6.35.2 PUT SMTP ... 2296
3.6.35.3 POST SMTP Test ... 2299
3.6.36 SSL ... 2303
3.6.36.1 GET SSL Parts ID ... 2305
3.6.36.2 GET SSL Endpoints ID ... 2307
3.6.36.3 DELETE SSL NetworkRanges ID ... 2308
3.6.36.4 GET SSL NetworkRanges ID ... 2309
3.6.36.5 GET SSL Networks Identifier ... 2310
3.6.36.6 GET SSL ... 2320
3.6.36.7 GET SSL Networks ... 2322
3.6.36.8 POST SSL Networks ... 2333
3.6.36.9 PUT SSL Networks ... 2347
3.6.36.10 GET SSL Endpoints ID History ... 2361

3.6.36.11 GET SSL Networks ID Parts ... 2366
3.6.36.12 POST SSL NetworkRanges ... 2368
3.6.36.13 PUT SSL NetworkRanges ... 2369
3.6.36.14 PUT SSL Endpoints Review Status ... 2370
3.6.36.15 PUT SSL Endpoints Monitor Status ... 2371
3.6.36.16 PUT SSL Endpoints Review All ... 2371
3.6.36.17 PUT SSL Endpoints Monitor All ... 2372
3.6.36.18 POST SSL Networks ID Scan ... 2373
3.6.36.19 POST SSL Networks ID Reset ... 2373
3.6.36.20 POST SSL NetworkRanges Validate ... 2374
3.6.36.21 DELETE SSL Networks ID ... 2375
3.6.37 Status ... 2375
3.6.37.1 GET Status Endpoints ... 2375
3.6.38 Templates ... 2376
3.6.38.1 GET Templates ID ... 2377
3.6.38.2 GET Templates Settings ... 2395
3.6.38.3 PUT Templates Settings ... 2402
3.6.38.4 GET Templates Subject Parts ... 2421
3.6.38.5 GET Templates ... 2422
3.6.38.6 PUT Templates ... 2435
3.6.38.7 POST Templates Import ... 2470
3.6.39 Workflow Certificates ... 2470
3.6.39.1 GET Workflow Certificates ID ... 2471
3.6.39.2 GET Workflow Certificates Denied ... 2475
3.6.39.3 GET Workflow Certificates Pending ... 2478
3.6.39.4 GET Workflow Certificates External Validation ... 2481
3.6.39.5 POST Workflow Certificates Deny ... 2484
3.6.39.6 POST Workflow Certificates Approve ... 2486
3.6.40 Workflow Definitions ... 2487
3.6.40.1 GET Workflow Definitions Steps Extension Name ... 2489
3.6.40.2 DELETE Workflow Definitions Definition ID ... 2494
3.6.40.3 GET Workflow Definitions Definition ID ... 2494
3.6.40.4 PUT Workflow Definitions Definition ID ... 2518
3.6.40.5 GET Workflow Definitions ... 2542
3.6.40.6 POST Workflow Definitions ... 2545
3.6.40.7 GET Workflow Definitions Steps ... 2570
3.6.40.8 GET Workflow Definitions Types ... 2576
3.6.40.9 PUT Workflow Definitions Definition ID Steps ... 2579
3.6.40.10 POST Workflow Definitions Definition ID Publish ... 2604
3.6.41 Workflow Instances ... 2628
3.6.41.1 DELETE Workflow Instances Instance Id ... 2629
3.6.41.2 GET Workflow Instances Instance ID ... 2630
3.6.41.3 GET Workflow Instances ... 2655
3.6.41.4 GET Workflow Instances My ... 2660
3.6.41.5 GET Workflow Instances AssignedToMe ... 2665
3.6.41.6 POST Workflow Instances Instance Id Stop ... 2670
3.6.41.7 POST Workflow Instances Instance ID Signals ... 2671
3.6.41.8 POST Workflow Instances Instance Id Restart ... 2673
3.7 API Change Log ... 2674
3.7.1 v9 API Change Logs ... 2674
3.7.1.1 API Change Log v9.0 ... 2674
3.7.1.2 API Change Log v9.1 ... 2676
3.7.1.3 API Change Log v9.2 ... 2677
3.7.1.4 API Change Log v9.3 ... 2678
3.7.1.5 API Change Log v9.4 ... 2678
3.7.1.6 API Change Log v9.5 ... 2678
3.7.1.7 API Change Log v9.6 ... 2679
3.7.1.8 API Change Log v9.7 ... 2679
3.7.1.9 API Change Log v9.8 ... 2679
3.7.1.10 API Change Log v9.9 ... 2679
3.7.2 v10 API Change Logs ... 2679
3.7.2.1 API Change Log v10.0 ... 2680

3.7.2.2 API Change Log v10.1 ... 2686
3.7.2.3 API Change Log v10.2 ... 2686
3.7.2.4 API Change Log v10.3.1 ... 2686
3.7.2.5 API Change Log v10.4 ... 2687
3.7.2.6 API Change Log v10.4.3 ... 2687
3.7.2.7 API Change Log v10.4.5 ... 2688
3.7.2.8 API Change Log v10.4.6 ... 2688
3.7.3 v11 API Change Logs ... 2689
3.7.3.1 API Change Log v11.0 ... 2689
4.0 Installing Servers ... 2695
4.1 Logical Architecture ... 2696
4.2 Physical Architecture ... 2699
4.3 Solution Design ... 2701
4.4 Keyfactor Command Server ... 2701
4.4.1 System Requirements ... 2702
4.4.2 Planning & Preparing ... 2704
4.4.2.1 Selecting an Identity Provider for Keyfactor Command ... 2704
4.4.2.2 SQL Server ... 2742
4.4.2.3 Certificate Authorities ... 2755
4.4.2.4 Keyfactor Command Server(s) ... 2756
4.4.2.5 Create Service Accounts for Keyfactor Command ... 2757
4.4.2.6 Create Groups to Control Access to Keyfactor Command Features ... 2762
4.4.2.7 Configure Certificate Chain Trusts for CAs ... 2763
4.4.2.8 Hostname Identification and Resolution ... 2764
4.4.2.9 Firewall Considerations ... 2765
4.4.2.10 Acquire a Public Key Certificate for the Keyfactor Command Server ... 2767
4.4.2.11 Install IIS and .NET on the Keyfactor Command Server ... 2768
4.4.2.12 Configure SSL for the Default Web Site on the Keyfactor Command Server ... 2774
4.4.2.13 Configure the Keyfactor Command Server to Require SSL ... 2774
4.4.2.14 Prepare for External Log Shipping over TLS (Optional) ... 2775
4.4.3 Installing ... 2780
4.4.3.1 Install the Keyfactor Command Components on the Keyfactor Command Server(s) ... 2780
4.4.3.2 Install the Keyfactor Command Server from the Command Line ... 2814
4.4.4 Initial Configuration ... 2821
4.4.4.1 Configure Kerberos Authentication ... 2822
4.4.4.2 Configure Logging ... 2829
4.4.4.3 Configure CA Certificate Synchronization ... 2830
4.4.4.4 Create or Identify Certificate Templates for Enrollment ... 2840
4.4.4.5 Configure Renewal Handler Permission ... 2841
4.4.4.6 Create a Certificate Template for Mac Auto-Enrollment ... 2843
4.5 Keyfactor CA Policy Module ... 2844
4.5.1 System Requirements ... 2845
4.5.2 Preparing for the Keyfactor CA Policy Module ... 2845
4.5.3 Installing the Keyfactor CA Policy Module Handlers ... 2846
4.5.3.1 Install the Keyfactor RFC 2818 Policy Handler ... 2850
4.5.3.2 Install the Keyfactor SAN Attribute Policy Handler ... 2855
4.5.3.3 Install the Keyfactor Whitelist Policy Handler ... 2861
4.5.4 Configure Logging for the Keyfactor CA Policy Module ... 2868
4.5.5 Add Non-Keyfactor SCEP Servers to the Ignore List ... 2870
4.6 Appendices ... 2870
4.6.1 Appendix - Troubleshooting Logi Log Files ... 2870
4.6.2 Appendix - Logi Load Balancing: Keyfactor Command Configuration Wizard Setup ... 2872
4.6.3 Appendix - Configuration Wizard Errors in the Logs ... 2874
5.0 Installing Orchestrators ... 2875
5.1 Orchestrator Job Overview ... 2877
5.2 Universal Orchestrator ... 2879
5.2.1 Preparing for the Universal Orchestrator ... 2880
5.2.1.1 System Requirements ... 2880
5.2.1.2 Create Service Accounts for the Universal Orchestrator ... 2884
5.2.1.3 Configure Certificate Root Trust for the Universal Orchestrator ... 2888

5.2.1.4 Grant the Orchestrator Service Account Permissions on the CAs ... 2889
5.2.1.5 Acquire a Certificate for Client Certificate Authentication (Optional) ... 2891
5.2.1.6 Upgrading the Universal Orchestrator ... 2895
5.2.2 Install the Universal Orchestrator on Windows ... 2898
5.2.3 Install the Universal Orchestrator on a Linux Server ... 2912
5.2.4 Install the Universal Orchestrator in a Linux Container ... 2922
5.2.5 Optional Configuration ... 2934
5.2.5.1 Configure Windows Targets for Remote Management ... 2935
5.2.5.2 Configure the Universal Orchestrator for Remote CA Management ... 2938
5.2.5.3 Installing Custom-Built Extensions ... 2940
5.2.5.4 Configuring Script-Based Certificate Store Jobs ... 2946
5.2.5.5 Configure Logging for the Universal Orchestrator ... 2950
5.2.5.6 Start the Universal Orchestrator Service ... 2953
5.2.5.7 Change Service Account Passwords ... 2954
5.2.5.8 Register a Client Certificate Renewal Extension ... 2961
5.3 Java Agent ... 2968
5.3.1 Preparing for the Java Agent ... 2968
5.3.1.1 Create Service Accounts for the Java Agent ... 2969
5.3.1.2 Create a Group for Java Agent Auto-Registration (Optional) ... 2970
5.3.1.3 Configure Certificate Root Trust for the Java Agent ... 2971
5.3.1.4 Create Environment Variables for the Java Agent on Windows ... 2971
5.3.2 Install the Java Agent on Windows ... 2974
5.3.3 Install the Java Agent on Linux ... 2979
5.3.4 Optional Configuration ... 2986
5.3.4.1 Configure Logging for the Java Agent ... 2986
5.3.4.2 Start the Keyfactor Java Agent Service ... 2989
5.3.4.3 Uninstall the Java Agent ... 2990
5.4 Bash Orchestrator ... 2991
5.4.1 Preparing for the Keyfactor Bash Orchestrator ... 2992
5.4.1.1 System Requirements ... 2992
5.4.1.2 Create a Service Account for the Keyfactor Bash Orchestrator ... 2994
5.4.1.3 Create a Group for Auto-Registration (Optional) ... 2994
5.4.1.4 Certificate Root Trust for the Keyfactor Bash Orchestrator ... 2995
5.4.2 Install the Keyfactor Bash Orchestrator ... 2995
5.4.3 Install Remote Control Targets ... 3000
5.4.4 Optional Configuration ... 3001
5.4.4.1 Configure Logging for the Keyfactor Bash Orchestrator ... 3002
5.4.4.2 Start the Keyfactor Bash Orchestrator Service ... 3003
5.5 Troubleshooting ... 3003
5.6 Appendices ... 3021
5.6.1 Appendix - Generate New Credentials for the Java Agent ... 3021
5.6.2 Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC ... 3023
5.6.3 Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication with Certificates Stored in Active Directory ... 3035
5.6.4 Appendix - Set up the Universal Orchestrator to Use a Forwarding Proxy ... 3049

6.0 Release Notes & Upgrading ... 3051

6.1 Upgrade Overviews ... 3051
6.1.1 Upgrade Overview - Keyfactor-Hosted ... 3051
6.1.1.1 Upgrading ... 3052
6.1.1.2 Post-Upgrade Steps ... 3054
6.1.2 Upgrade Overview - Self-Hosted ... 3055
6.1.2.1 Preparing ... 3056
6.1.2.2 Upgrading ... 3066
6.1.2.3 Post-Upgrade Steps ... 3071
6.1.2.4 Troubleshooting ... 3073
6.2 Major Release 11.0 Notes ... 3076
6.2.1 Incremental Release 11.1 Notes ... 3092
6.3 Major Release 10.0 Notes ... 3094
6.3.1 Incremental Release 10.5 Notes ... 3111
6.3.2 Hot Fix Release 10.4.6 Notes ... 3112

6.3.3 Hot Fix Release 10.4.5 Notes 3114
6.3.4 Hot Fix Release 10.4.4 Notes 3115
6.3.5 Hot Fix Release 10.4.3 Notes 3116
6.3.6 Hot Fix Release 10.4.2 Notes 3117
6.3.7 Hot Fix Release 10.4.1 Notes 3118
6.3.8 Incremental Release 10.4 Notes 3119
6.3.9 Hot Fix Release 10.3.1 Notes 3123
6.3.10 Incremental Release 10.3 Notes 3124
6.3.11 Incremental Release 10.2 Notes 3126
6.3.12 Incremental Release 10.1 Notes 3129
6.4 Major Release 9.0 Notes 3130
6.4.1 Incremental Release 9.10 Notes 3145
6.4.2 Incremental Release 9.9 Notes 3146
6.4.3 Incremental Release 9.8 Notes 3147
6.4.4 Incremental Release 9.7 Notes 3148
6.4.5 Incremental Release 9.6 Notes 3150
6.4.6 Incremental Release 9.5 Notes 3151
6.4.7 Incremental Release 9.4 Notes 3153
6.4.8 Incremental Release 9.3 Notes 3155
6.4.9 Incremental Release 9.2 Notes 3156
6.4.10 Incremental Release 9.1 Notes 3160
6.5 Keyfactor Command v11 Compatibility Matrix 3164
6.6 Keyfactor Command v10 Compatibility Matrix 3167
6.7 Keyfactor Command v9 Compatibility Matrix 3170
7.0 Glossary 3175
8.0 Copyright Notice 3185

List of Tables

Table 1: Status Tab Descriptions 25
Table 2: Validation Tab Descriptions 28
Table 3: Chart of Available Exports per Standard Report 92
Table 4: Substitutable Special Text for Expiration Alerts 174
Table 5: Substitutable Special Text for Pending Request Alerts 186
Table 6: Substitutable Special Text for Issued Certificate Alerts 194
Table 7: Substitutable Special Text for Denied Certificate Request Alerts 201
Table 8: Substitutable Special Text for Key Rotation Alerts 210
Table 9: PowerShell Event Handler Special Fields 220
Table 10: PowerShell Event Handler Special Fields 225
Table 11: Tokens for Workflow Definitions 296
Table 12: CA Function Matrix 350
Table 13: Supported Regular Expressions for Enrollment with Examples 403
Table 14: Discovery Options 452
Table 15: SSL Email Notification Values Defined 480
Table 16: Orchestrator Capabilities 483
Table 17: SSH Permissions Table 598
Table 18: Console Application Settings 603
Table 19: Audit Log Application Settings 608
Table 20: Enrollment Application Settings 610
Table 21: Agents Application Settings 615
Table 22: API Application Settings 619
Table 23: SSH Application Settings 620
Table 24: Workflow Application Settings 621
Table 25: Agents Security Role Permissions v2 633
Table 26: Application Settings Security Role Permissions v2 636
Table 27: Auditing Security Role Permissions v2 636
Table 28: Certificate Authorities Security Role Permissions v2 636
Table 29: Certificate Stores Security Role Permissions v2 637
Table 30: Certificate Templates Security Role Permissions v2 646
Table 31: Certificates Security Role Permissions v2 646
Table 32: Dashboard Security Role Permissions v2 659
Table 33: Identity Providers Security Role Permissions v2 660
Table 34: Certificate Metadata Types Security Role Permissions v2 660
Table 35: Monitoring Security Role Permissions v2 661
Table 36: Privileged Access Management Security Role Permissions v2 663
Table 37: Management Portal Security Role Permissions v2 663
Table 38: Reports Security Role Permissions v2 664
Table 39: Scripts Security Role Permissions v2 665
Table 40: Security Settings Security Role Permissions v2 665
Table 41: SSH Security Role Permissions v2 666
Table 42: SSL Management Security Role Permissions v2 666
Table 43: System Settings Permissions v2 667
Table 44: Workflows Security Role Permissions v2 668
Table 45: Agent Auto-Registration Security Role Permissions v1 670
Table 46: Agent Management Security Role Permissions v1 671
Table 47: Alerts Security Role Permissions v1 671
Table 48: Application Settings Security Role Permissions v1 671
Table 49: Auditing Security Role Permissions v1 672
Table 50: Certificate Collections Security Role Permissions v1 672
Table 51: Certificate Enrollment Security Role Permissions v1 672
Table 52: Certificate Metadata Types Security Role Permissions v1 673
Table 53: Certificate Requests Security Role Permissions v1 673

Table 54: Certificate Store Management Security Role Permissions v1 673
Table 55: Certificates Security Role Permissions v1 674
Table 56: Dashboard Security Role Permissions v1 677
Table 57: Event Handler Registration Security Role Permissions v1 677
Table 58: Identity Providers Security Role Permissions v1 677
Table 59: Mac Auto-Enroll Management Security Role Permissions v1 677
Table 60: Management Portal Security Role Permissions v1 678
Table 61: Monitoring Security Role Permissions v1 678
Table 62: PKI Management Security Role Permissions v1 678
Table 63: Privileged Access Management Security Role Permissions v1 679
Table 64: Reports Security Role Permissions v1 679
Table 65: Scripts Security Role Permissions v1 680
Table 66: Security Settings Security Role Permissions v1 680
Table 67: SSH Security Role Permissions v1 680
Table 68: SSL Management Security Role Permissions v1 681
Table 69: System Settings Security Role Permissions v1 681
Table 70: Workflow Definitions Security Role Permissions v1 682
Table 71: Workflow Instances Security Role Permissions v1 682
Table 72: Certificate Metadata Data Type Dialog Options 714
Table 73: Audit Download CSV Records 721
Table 74: Audit Operations 728
Table 75: Audit Categories 729
Table 76: Identity Provider Parameters 756
Table 77: Appsetting.json File Parameters - WebAgentServices 774
Table 78: Appsetting.json File Parameters - ClaimsProxy 776
Table 79: Appsetting.json File Parameters - KeyfactorAPI 778
Table 80: Keyfactor Command Services Configuration Settings 781
Table 81: Keyfactor Command Jobs Services 782
Table 82: Appsetting.json File Parameters - WebPortalServices 786
Table 83: Appsetting.json file - SQL Retry Parameters 788
Table 84: Bash Orchestrator Job History Warning Resolution 791
Table 85: NLog.config Files for Keyfactor Command 803
Table 86: Keyfactor Command Windows Event IDs 806
Table 87: Keyfactor Command Windows Event IDs for Audit Log 814
Table 88: Keyfactor Universal Orchestrator Windows Event IDs 816
Table 89: Third-Party Notices for Keyfactor Command Software Distributions 835
Table 90: Common Request Headers 850
Table 91: Common Response Headers 850
Table 92: HTTP Statuses 851
Table 93: Agents Endpoints 853
Table 94: GET Agents{id} Input Parameters 854
Table 95: GET Agent {id} Response Data 855
Table 96: GET Agents Input Parameters 859
Table 97: GET Agent Response Data 861
Table 98: POST Agents Reset Input Parameters 864
Table 99: POST Agents Approve Input Parameters 865
Table 100: POST Agents Disapprove Input Parameters 865
Table 101: POST Agents {id} Reset Input Parameters 866
Table 102: POST Agents {id} FetchLogs Input Parameters 867
Table 103: POST Agents Set Auth Certificate Reenrollment Input Parameters 868
Table 104: POST Agents Set Auth Certificate Reenrollment Response Data 869
Table 105: Agent BluePrint Endpoints 869
Table 106: DELETE Agent BluePrint {id} Input Parameters 870
Table 107: GET Agent BluePrint {id} Input Parameters 871
Table 108: GET Agent BluePrint {id} Response Data 871
Table 109: GET Agent BluePrint Input Parameters 872
Table 110: GET Agent BluePrint Response Data 873

Table 111: GET Agent BluePrint {id} Jobs Input Parameters 874
Table 112: GET Agent BluePrint {id} Jobs Response Data 875
Table 113: GET Agent BluePrint {id} Stores Input Parameters 879
Table 114: GET Agent BluePrint {id} Stores Response Data 880
Table 115: POST Agent BluePrint Apply BluePrint Input Parameters 881
Table 116: POST Agent BluePrint Generate Input Parameters 882
Table 117: POST Agent BluePrint Generate Response Data 882
Table 118: Agent Pool Endpoints 883
Table 119: DELETE Agent Pools {id} Input Parameters 884
Table 120: GET Agent Pools {id} Input Parameters 884
Table 121: GET AgentPools {id} Response Data 885
Table 122: GET Agent Pools Input Parameters 887
Table 123: GET AgentPools Response Data 888
Table 124: POST Agent Pools Input Parameters 889
Table 125: POST Agent Pools Response Data 890
Table 126: PUT Agent Pools Input Parameters 892
Table 127: PUT Agent Pools Response Data 893
Table 128: GET Agent Pools Default Agent Pool Agents Input Parameters 895
Table 129: GET Agent Pools Default Agent Pool Agents Response Data 896
Table 130: Alerts Denied 897
Table 131: DELETE Alerts Denied {id} Input Parameters 897
Table 132: GET Alerts Denied {id} Input Parameters 898
Table 133: GET Alerts Denied {id} Response Data 899
Table 134: GET Alerts Denied Input Parameters 903
Table 135: GET Alerts Denied Response Data 904
Table 136: POST Alerts Denied Input Parameters 909
Table 137: POST Alerts Denied Response Data 914
Table 138: PUT Alerts Denied Input Parameters 919
Table 139: PUT Alerts Denied Response Data 924
Table 140: Alerts Expiration 928
Table 141: DELETE Alerts Expiration {id} Input Parameters 929
Table 142: GET Alerts Expiration {id} Input Parameters 929
Table 143: GET Alerts Expiration {id} Response Data 930
Table 144: GET Alerts Expiration Schedule Response Data 934
Table 145: PUT Alerts Expiration Schedule Input Parameters 935
Table 146: PUT Alerts Expiration Schedule Response Data 936
Table 147: GET Alerts Expiration Input Parameters 937
Table 148: GET Alerts Expiration Response Data 938
Table 149: POST Alerts Expiration Input Parameters 943
Table 150: POST Alerts Expiration Response Data 948
Table 151: PUT Alerts Expiration Input Parameters 953
Table 152: PUT Alerts Expiration Response Data 958
Table 153: POST Alerts Expiration Test Input Parameters 962
Table 154: POST Alerts Expiration Test Response Data 963
Table 155: POST Alerts Expiration Test All Input Parameters 964
Table 156: POST Alerts Expiration Test All Response Data 965
Table 157: Alerts Issued 966
Table 158: DELETE Alerts Issued {id} Input Parameters 967
Table 159: GET Alerts Issued {id} Input Parameters 967
Table 160: GET Alerts Issued {id} Response Data 968
Table 161: GET Alerts Issued Schedule Response Data 972
Table 162: PUT Alerts Issued Schedule Input Parameters 974
Table 163: PUT Alerts Issued Schedule Response Data 975
Table 164: GET Alerts Issued Input Parameters 977
Table 165: GET Alerts Issued Response Data 978
Table 166: POST Alerts Issued Input Parameters 983
Table 167: POST Alerts Issued Response Data 988

Table 168: PUT Alerts Issued Input Parameters 993
Table 169: PUT Alerts Issued Response Data 998
Table 170: Alerts Key Rotation 1002
Table 171: DELETE Alerts Key Rotation {id} Input Parameters 1003
Table 172: GET Alerts Key Rotation {id} Input Parameters 1003
Table 173: GET Alerts Key Rotation {id} Response Data 1004
Table 174: GET Alerts Key Rotation Schedule Response Data 1007
Table 175: PUT Alerts Key Rotation Schedule Input Parameters 1009
Table 176: PUT Alerts Key Rotation Schedule Response Data 1010
Table 177: GET Alerts Key Rotation Input Parameters 1012
Table 178: GET Alerts Key Rotation Response Data 1013
Table 179: POST Alerts Key Rotation Input Parameters 1016
Table 180: POST Alerts Key Rotation Response Data 1020
Table 181: PUT Alerts Key Rotation Input Parameters 1024
Table 182: PUT Alerts Key Rotation Response Data 1028
Table 183: POST Alerts Key Rotation Test Input Parameters 1031
Table 184: POST Alerts Key Rotation Test Response Data 1032
Table 185: POST Alerts Key Rotation Test All Input Parameters 1033
Table 186: POST Alerts Key Rotation Test All Response Data 1034
Table 187: Alerts Pending 1035
Table 188: DELETE Alerts Pending {id} Input Parameters 1036
Table 189: GET Alerts Pending {id} Input Parameters 1036
Table 190: GET Alerts Pending {id} Response Data 1037
Table 191: GET Alerts Pending Schedule Response Data 1041
Table 192: PUT Alerts Pending Schedule Input Parameters 1043
Table 193: PUT Alerts Pending Schedule Response Data 1044
Table 194: GET Alerts Pending Input Parameters 1046
Table 195: GET Alerts Pending Response Data 1047
Table 196: POST Alerts Pending Input Parameters 1052
Table 197: POST Alerts Pending Response Data 1057
Table 198: PUT Alerts Pending Input Parameters 1062
Table 199: PUT Alerts Pending Response Data 1067
Table 200: POST Alerts Pending Test Input Parameters 1071
Table 201: POST Alerts Pending Test Response Data 1072
Table 202: POST Alerts Pending Test All Input Parameters 1073
Table 203: POST Alerts Pending Test All Response Data 1074
Table 204: AppSetting Endpoints 1075
Table 205: GET AppSetting Response Data 1076
Table 206: GET AppSetting {id} Input Parameters 1077
Table 207: GET AppSetting {id} Response Data 1078
Table 208: PUT AppSetting Input Parameters 1079
Table 209: PUT AppSetting Response Data 1080
Table 210: PUT AppSetting {id} Set Input Parameters 1081
Table 211: PUT AppSetting {id} Set Response Data 1082
Table 212: PUT AppSetting {name} Set Input Parameters 1083
Table 213: PUT AppSetting {name} Set Response Data 1084
Table 214: Audit Endpoints 1085
Table 215: GET Audit {id} Input Parameters 1085
Table 216: GET Audit {id} Response Data 1086
Table 217: GET Audit {id} Validate Input Parameters 1090
Table 218: GET Audit {id} Validate Response Data 1091
Table 219: GET Audit Input Parameters 1092
Table 220: GET Audit Response Data 1093
Table 221: GET Audit Download Input Parameters 1098
Table 222: GET Audit Download Response Data 1099
Table 223: GET Audit Related Entities Input Parameters 1102
Table 224: GET Audit Related Entities Response Data 1103

Table 225: Certificates Endpoints 1107
Table 226: GET Certificates {id} Security Input Parameters 1110
Table 227: GET Certificates {id} Security Response Data 1110
Table 228: GET Certificates {id} Validate Input Parameters 1111
Table 229: GET Certificates {id} Validate Response Data 1112
Table 230: GET Certificates Locations {id} Input Parameters 1117
Table 231: GET Certificates Locations {id} Response Data 1118
Table 232: GET Certificates Identity Audit {id} v2 Input Parameters 1120
Table 233: GET Certificates Identity Audit {id} v2 Response Data 1121
Table 234: GET Certificates Identity Audit {id} Iv1 nput Parameters 1122
Table 235: GET Certificates Identity Audit {id} v1 Response Data 1123
Table 236: DELETE Certificates {id} Input Parameters 1124
Table 237: GET Certificates {id} Input Parameters 1125
Table 238: GET Certificates {id} Response Data 1126
Table 239: GET Certificates Metadata Compare Input Parameters 1137
Table 240: GET Certificates {id} History Input Parameters 1139
Table 241: GET Certificates {id} History Response Data 1140
Table 242: DELETE Certificates Input Parameters 1141
Table 243: GET Certificates Input Parameters 1143
Table 244: GET Certificates Response Data 1146
Table 245: PUT Certificates Metadata Input Parameters 1157
Table 246: PUT Certificates Metadata All Input Parameters 1158
Table 247: POST Certificates Import Input Parameters 1163
Table 248: POST Certificates Import Response Data 1165
Table 249: POST Certificates Revoke Input Parameters 1167
Table 250: POST Certificates Analyze Input Parameters 1168
Table 251: POST Certificates Analyze Response Data 1169
Table 252: POST Certificates Recover Input Parameters 1171
Table 253: POST Certificates Recover Response Data 1173
Table 254: POST Certificates Download Input Parameters 1175
Table 255: POST Certificates Download Response Data 1176
Table 256: POST Certificates Revoke All Input Parameters 1178
Table 257: DELETE Certificates Query Input Parameters 1180
Table 258: DELETE Certificates Private Key Input Parameters 1181
Table 259: DELETE Certificates Private Key {id} Input Parameters 1182
Table 260: Certificate Authority Endpoints 1182
Table 261: DELETE Certificate Authority {id} Input Parameters 1186
Table 262: GET Certificate Authority {id} Input Parameters 1186
Table 263: GET Certificate Authority {id} Response Data 1187
Table 264: GET Certificate Authority Input Parameters 1201
Table 265: GET Certificate Authority Response Data 1202
Table 266: POST Certificate Authority Input Parameters 1217
Table 267: POST Certificate Authority Response Data 1234
Table 268: PUT Certificate Authority Input Parameters 1249
Table 269: PUT Certificate Authority Response Data 1266
Table 270: POST Certificate Authority Test Input Parameters 1281
Table 271: POST Certificate Authority Test Response Data 1283
Table 272: POST Certificate Authority PublishCRL Input Parameters 1284
Table 273: GET Certificate Authority Source Count Response Body 1285
Table 274: GET Certificate Authority Available Forests Response Body 1285
Table 275: GET Certificate Authority Health Monitoring Schedule Response Body 1286
Table 276: GET Certificate Authority Alert Recipients CA Health Recipients Response Body 1287
Table 277: POST Certificate Authority Alert Recipients CA Health Recipients Input Body 1287
Table 278: POST Certificate Authority Alert Recipients CA Health Recipients Response Body 1288
Table 279: GET Certificate Authority Alert Recipients CA Health Recipients {id} Input Parameters 1288
Table 280: GET Certificate Authority Alert Recipients CA Health Recipients {id} Response Body 1289
Table 281: DELETE Certificate Authority Alert Recipients CA Health Recipients {id} Input Parameters 1289

Table 282: PUT Certificate Authority Alert Recipients CA Health Recipients {id} Input Body 1290
Table 283: PUT Certificate Authority Alert Recipients CA Health Recipients {id} Response Body 1290
Table 284: DELETE Certificate Authority Alert Recipients CA Threshold Recipient {id} Input Parameters 1291
Table 285: GET Certificate Authority Alert Recipients CA Threshold Recipients Response Body 1291
Table 286: GET Certificate Authority Alert Recipients CA Threshold Recipients {id} Input Parameters 1292
Table 287: GET Certificate Authority Alert Recipients CA Threshold Recipients {id} Response Body 1292
Table 288: POST Certificate Authority Alert Recipients CA Threshold Recipients Input Body 1293
Table 289: POST Certificate Authority Alert Recipients CA Threshold Recipients Response Body 1293
Table 290: PUT Certificate Authority Alert Recipients CA Threshold Recipients {id} Input Body 1294
Table 291: PUT Certificate Authority Alert Recipients CA Threshold Recipients {id} Response Body 1294
Table 292: POST Certificate Authority Import Input Parameters 1294
Table 293: Certificate Collections Endpoints 1295
Table 294: GET Certificate Collections Input Parameters 1297
Table 295: GET Certificate Collections Response Data 1298
Table 296: POST Certificate Collections Input Parameters 1300
Table 297: POST Certificate Collections Response Data 1304
Table 298: PUT Certificate Collections Input Parameters 1306
Table 299: PUT Certificate Collections Response Data 1308
Table 300: GET Certificate Collections {id} Input Parameters 1309
Table 301: GET Certificate Collections {id} Response Data 1310
Table 302: DELETE Certificate Collection {id} Input Parameters 1311
Table 303: GET Certificate Collections Name Input Parameters 1312
Table 304: GET Certificate Collections ID Response Data 1313
Table 305: POST Certificate Collections Copy Input Parameters 1315
Table 306: POST Certificate Collections Copy Response Data 1319
Table 307: GET Certificate Collection Nav Items Response Data 1320
Table 308: PUT Certificate Collection{id} Favorite Input Body 1321
Table 309: GET Certificate Collections List Input Parameters 1322
Table 310: GET Certificate Collections List Response Data 1323
Table 311: Certificate Stores Endpoints 1324
Table 312: DELETE Certificate Stores Input Parameters 1326
Table 313: GET Certificate Stores Input Parameters 1328
Table 314: GET Certificate Stores Response Data 1330
Table 315: POST Certificate Stores Input Parameters 1340
Table 316: POST Certificate Stores Response Data 1358
Table 317: PUT Certificate Stores Input Parameters 1369
Table 318: PUT Certificate Stores Response Data 1388
Table 319: DELETE Certificate Stores Input Parameters 1398
Table 320: GET Certificate Stores {id} Input Parameters 1398
Table 321: GET Certificate Stores {id} Response Data 1399
Table 322: GET Certificate Stores {id} Inventory Input Parameters 1416
Table 323: GET Certificate Stores {id} Inventory Response Data 1417
Table 324: GET Certificate Stores Server Input Parameters 1419
Table 325: GET Certificate Stores Server Response Data 1420
Table 326: POST Certificate Stores Server Input Parameters 1422
Table 327: POST Certificate Stores Server Response Data 1426
Table 328: PUT Certificate Stores Server Input Parameters 1428
Table 329: PUT Certificate Stores Server Response Data 1431
Table 330: PUT Certificate Stores Password Input Parameters 1433
Table 331: PUT Certificate Stores Discovery Job Input Parameters 1436
Table 332: PUT Certificate Stores Assign Container Input Parameters 1441
Table 333: PUT Certificate Stores Assign Container Response Data 1442
Table 334: POST Certificate Stores Approve Input Parameters 1452
Table 335: POST Certificate Stores Schedule Input Parameters 1463
Table 336: POST Certificates Stores Reenrollment Input Parameters 1466
Table 337: POST Certificate Stores Certificates Add Input Parameters 1468
Table 338: POST Certificate Stores Certificates Remove Input Parameters 1474

Table 339: Certificate Store Containers Endpoints 1476
Table 340: GET Certificate Store Containers Input Parameters 1478
Table 341: GET Certificate Stores Containers Response Data 1479
Table 342: POST Certificate Stores Containers Input Parameters 1481
Table 343: POST Certificate Stores Containers Response Data 1483
Table 344: PUT Certificate Store Containers Input Parameters 1486
Table 345: PUT Certificate Store Containers Response Data 1488
Table 346: DELETE Certificate Store Containers {id} Input Parameters 1509
Table 347: GET Certificate Store Containers {id} Input Parameters 1510
Table 348: GET Certificate Stores Containers {id} Response Data 1511
Table 349: Certificate Store Type Endpoints 1531
Table 350: DELETE Certificate Store Types {id} Input Parameters 1532
Table 351: GET Certificate Store Types {id} Input Parameters 1532
Table 352: GET Certificate Store Types {id} Response Data 1533
Table 353: GET Certificate Store Types Name {Name} Input Parameters 1539
Table 354: GET Certificate Store Types Name {Name} Response Data 1540
Table 355: DELETE Certificate Store Types Input Parameters 1546
Table 356: GET Certificate Store Types Input Parameters 1546
Table 357: GET Certificate Store Types Response Data 1547
Table 358: POST Certificate Store Types Input Parameters 1553
Table 359: POST Certificate Store Types Response Data 1561
Table 360: PUT Certificate Store Types Input Parameters 1568
Table 361: PUT Certificate Store Types Response Data 1577
Table 362: Component Installation Endpoints 1582
Table 363: DELETE Component Installation {id} Input Parameters 1583
Table 364: GET Component Installation Input Parameters 1584
Table 365: GET Component Installation Response Data 1585
Table 366: CSR Generation Endpoints 1586
Table 367: DELETE CSR Generation Pending {id} Input Parameters 1586
Table 368: GET CSR Generation Pending {id} Input Parameters 1587
Table 369: GET CSR Generation Pending {id} Response Data 1587
Table 370: DELETE CSR Generation Pending Input Parameters 1588
Table 371: GET CSR Generation Pending Input Parameters 1589
Table 372: GET CSR Generation Pending Response Data 1589
Table 373: POST CSR Generation Generate Input Parameters 1591
Table 374: POST CSR Generation Generate Response Data 1594
Table 375: Custom Job Types Endpoints 1595
Table 376: DELETE JobTypes Custom {id} Input Parameters 1595
Table 377: GET JobTypes Custom {id} Input Parameters 1596
Table 378: GET JobTypes Custom {id} Response Data 1597
Table 379: GET Job Types Custom Input Parameters 1598
Table 380: GET Job Types Custom Response Data 1599
Table 381: POST JobTypes Custom Input Parameters 1601
Table 382: POST JobTypes Custom Response Data 1603
Table 383: PUT JobTypes Custom Input Parameters 1605
Table 384: PUT JobTypes Custom Response Data 1607
Table 385: Enrollment Endpoints 1608
Table 386: GET Enrollment Settings {id} Input Parameters 1609
Table 387: GET Enrollment Settings {id} Response Data 1610
Table 388: GET Enrollment CSR Content My Response Data 1617
Table 389: GET Enrollment PFX Content My Response Data 1636
Table 390: GET Enrollment Available Renewal ID {id} Input Parameters 1654
Table 391: GET Enrollment Available Renewal ID {id} Response Data 1655
Table 392: GET Enrollment Available Renewal Thumbprint {thumbprint} Input Parameters 1656
Table 393: GET Enrollment Available Renewal Thumbprint {thumbprint} Response Data 1657
Table 394: POST Enrollment CSR Input Parameters 1659
Table 395: POST Enrollment CSR Response Data 1663

Table 396: POST Enrollment PFX v2 Input Parameters 1666
Table 397: POST Enrollment PFX v2 Response Data 1673
Table 398: POST Enrollment PFX v1 Input Parameters 1676
Table 399: POST Enrollment PFX v1 Response Data 1681
Table 400: POST Enrollment CSR Parse Input Parameters 1683
Table 401: POST Enrollment CSR Parse Response Data 1684
Table 402: POST Enrollment PFX Deploy Input Parameters 1686
Table 403: POST Enrollment PFX Deploy Response Data 1691
Table 404: POST Enrollment PFX Replace Input Parameters 1693
Table 405: POST Enrollment PFX Replace Response Data 1694
Table 406: POST Enrollment Renew Input Parameters 1696
Table 407: POST Enrollment Renew Response Data 1697
Table 408: EventHandlerRegistration Endpoints 1698
Table 409: GET Event Handler Registration Input Parameters 1699
Table 410: GET Event Handler Registration Response Data 1700
Table 411: POST Event Handler Registration Input Parameters 1700
Table 412: POST Event Handler Registration Response Data 1701
Table 413: GET Event Handler Registration {id} Input Parameters 1702
Table 414: GET Event Handler Registration Response Data 1702
Table 415: PUT Event Handler Registration {id} Input Parameters 1703
Table 416: PUT Event Handler Registration {id} Response Data 1703
Table 417: DELETE Event Handler Registration {id} Input Parameters 1704
Table 418: Extensions Scripts Endpoints 1705
Table 419: DELETE Extensions Scripts Input Parameters 1705
Table 420: GET Extensions Scripts {id} Input Parameters 1706
Table 421: GET Extensions Scripts {id} Response Data 1706
Table 422: GET Extensions Scripts Input Parameters 1708
Table 423: GET Extensions Scripts Response Data 1709
Table 424: POST Extensions Scripts Input Parameters 1710
Table 425: POST Extensions Scripts Response Data 1713
Table 426: PUT Extensions Scripts Input Parameters 1714
Table 427: PUT Extensions Scripts Response Data 1715
Table 428: Identity Providers Endpoint 1715
Table 429: GET Identity Providers{id} Input Parameters 1716
Table 430: GET Identity Providers {id} Response Data 1717
Table 431: Identity Provider Parameters 1717
Table 432: Identity Provider Parameter Structure 1729
Table 433: PUT Identity Providers {id} Input Parameters 1732
Table 434: Identity Provider Parameters 1732
Table 435: Identity Provider Parameter Structure 1745
Table 436: PUT Identity Providers {id} Response Data 1747
Table 437: Identity Provider Parameters 1747
Table 438: Identity Provider Parameter Structure 1759
Table 439: GET Identity Providers Input Parameters 1761
Table 440: GET Identity Provider Response Data 1762
Table 441: Identity Provider Parameters 1762
Table 442: Identity Provider Parameter Structure 1774
Table 443: GET Identity Providers Types Response Data 1776
Table 444: License Endpoint 1777
Table 445: GET License Response Data 1778
Table 446: MacEnrollment Endpoints 1780
Table 447: GET MacEnrollment Response Data 1781
Table 448: PUT MacEnrollment input Parameters 1782
Table 449: PUT MacEnrollment Response Data 1783
Table 450: MetadataFields Endpoints 1783
Table 451: DELETE Metadata Fields {id} Input Parameters 1785
Table 452: GET Metadata Fields {id} Input Parameters 1785

Table 453: GET Metadata Fields {id} Response Data 1786
Table 454: GET Metadata Fields {name} Input Parameters 1789
Table 455: GET Metadata Fields {name} Response Data 1790
Table 456: GET Metadata Fields {id} In Use Input Parameters 1793
Table 457: GET Metadata Fields {id} In Use Response Data 1793
Table 458: DELETE Metadata Fields Input Parameters 1794
Table 459: GET Metadata Fields Input Parameters 1795
Table 460: GET Metadata Fields Response Data 1796
Table 461: POST Metadata Fields Input Parameters 1799
Table 462: POST Metadata Fields Response Data 1802
Table 463: PUT Metadata Fields Input Parameters 1806
Table 464: PUT Metadata Fields Response Data 1809
Table 465: Monitoring Endpoints 1811
Table 466: DELETE Monitoring Revocation {id} Input Parameters 1813
Table 467: GET Monitoring Revocation {id} Input Parameters 1813
Table 468: GET Monitoring Revocation {id} Response Data 1814
Table 469: GET Monitoring Revocation Input Parameters 1818
Table 470: GET Monitoring Revocation Response Data 1819
Table 471: POST Monitoring Revocation Input Parameters 1823
Table 472: POST Monitoring Revocation Response Data 1827
Table 473: PUT Monitoring Revocation {id} Input Parameters 1831
Table 474: PUT Monitoring Revocation {id} Response Data 1835
Table 475: POST Monitoring Resolve OCSP Input Parameters 1838
Table 476: POST Monitoring Resolve OCSP Response Data 1839
Table 477: POST Monitoring Revocation Test Input Parameters 1840
Table 478: POST Monitoring Revocation Test Response Data 1840
Table 479: POST Monitoring Revocation Test All Input Parameters 1841
Table 480: POST Monitoring Revocation Test All Response Data 1842
Table 481: Orchestrator Jobs Endpoints 1842
Table 482: GET Orchestrator Jobs Job Status Data Input Parameters 1844
Table 483: GET Orchestrator Jobs Job Status Data Response Data 1844
Table 484: GET Orchestrator Jobs Job History Input Parameters 1845
Table 485: GET Orchestrator Jobs Job History Response Data 1846
Table 486: GET Orchestrator Jobs Scheduled Jobs Input Parameters 1851
Table 487: GET Orchestrator Jobs Scheduled Jobs Response Data 1852
Table 488: POST Orchestrator Jobs Custom Input Parameters 1855
Table 489: POST Orchestrator Jobs Custom Response Data 1859
Table 490: POST Orchestrator Jobs Reschedule Input Parameters 1861
Table 491: POST Orchestrator Jobs Unschedule Input Parameters 1863
Table 492: POST Orchestrator Jobs Acknowledge Input Parameters 1864
Table 493: POST Orchestrator Jobs Custom Bulk Input Parameters 1866
Table 494: POST Orchestrator Jobs Custom Bulk Response Data 1870
Table 495: PamProviders Endpoints 1871
Table 496: DELETE PamProviders {id} v1 & v2 Input Parameters 1872
Table 497: GET PamProviders {id} v2 Input Parameters 1873
Table 498: GET PamProviders {id} v2 Response Data 1874
Table 499: GET PamProviders {id} Iv1 nput Parameters 1880
Table 500: GET PamProviders {id} v1 Response Data 1881
Table 501: GET PamProviders Types v1 & v2 Response Data 1889
Table 502: POST PamProviders Types v1 & v2 Input Parameters 1893
Table 503: POST PamProviders Types v1 & v2 Response Data 1896
Table 504: GET PamProviders v2 Input Parameters 1900
Table 505: GET PamProviders v2 Response Data 1901
Table 506: GET PamProviders v1 Input Parameters 1908
Table 507: GET PamProviders v1 Response Data 1909
Table 508: POST PamProviders v2 Input Parameters 1917
Table 509: POST PamProviders v2 Response Data 1921

Table 510: POST PamProviders v1 Input Parameters 1926
Table 511: POST PamProviders v2 Response Data 1930
Table 512: PUT PamProviders v2 Input Parameters 1936
Table 513: PUT PamProviders v2 Response Data 1940
Table 514: PUT PamProviders v1 Input Parameters 1945
Table 515: PUT PamProviders v1 Response Data 1949
Table 516: GET PamProviders Types {id} v2 Input Parameters 1954
Table 517: GET PamProviders Types {id} v2 Response Data 1955
Table 518: Security Roles Endpoints 1958
Table 519: GET Permissions Response Data 1958
Table 520: Permission Sets Endpoints 1961
Table 521: GET Permission Sets{id} Input Parameters 1961
Table 522: GET Permission Sets {id} Response Data 1962
Table 523: DELETE Permission Sets{id} Input Parameters 1962
Table 524: GET Permission Sets Input Parameters 1963
Table 525: GET Permission Sets Response Data 1963
Table 526: POST Permission Sets Input Parameters 1964
Table 527: POST Permission Sets Response Data 1965
Table 528: PUT Permission Sets Input Parameters 1966
Table 529: PUT Permission Sets Response Data 1966
Table 530: Reports Endpoints 1967
Table 531: GET Reports {id} Input Parameters 1968
Table 532: GET Reports {id} Response Data 1969
Table 533: DELETE Reports Custom {id} Input Parameters 1977
Table 534: GET Reports Custom {id} Input Parameters 1977
Table 535: GET Reports Custom {id} Response Data 1978
Table 536: DELETE Reports Schedules {id} Input Parameters 1979
Table 537: GET Reports Schedules {id} Input Parameters 1979
Table 538: GET Reports Schedules {id} Response Data 1980
Table 539: GET Reports {id} Parameters Input Parameters 1984
Table 540: GET Reports {id} Parameters Response Data 1985
Table 541: PUT Reports {id} Parameters Input Parameters 1986
Table 542: PUT Reports {id} Parameters Response Data 1987
Table 543: GET Reports Input Parameters 1989
Table 544: GET Reports Response Data 1990
Table 545: PUT Reports Input Parameters 1992
Table 546: PUT Reports Response Data 1993
Table 547: GET Reports Custom Input Parameters 1995
Table 548: GET Reports Custom Response Data 1996
Table 549: POST Reports Custom Input Parameters 1997
Table 550: POST Reports Custom Response Data 1997
Table 551: PUT Reports Custom Input Parameters 1998
Table 552: PUT Reports Custom Response Data 1999
Table 553: GET Reports {id} Schedules Input Parameters 2000
Table 554: GET Reports {id} Schedules Response Data 2001
Table 555: POST Reports {id} Schedules Input Parameters 2005
Table 556: POST Reports {id} Schedules Response Data 2011
Table 557: PUT Reports {id} Schedules Input Parameters 2015
Table 558: PUT Reports {id} Schedules Response Data 2021
Table 559: Scheduling Endpoints 2024
Table 560: POST Scheduling Input Parameters 2026
Table 561: POST Scheduling Response Data 2027
Table 562: Security Endpoints 2028
Table 563: DELETE Security Identities {id} Input Parameters 2029
Table 564: GET Security Identities {id} Input Parameters 2030
Table 565: GET Security Identities {id} Response Data 2031
Table 566: GET Security Identities Lookup Input Parameters 2034

Table 567: GET Security Identities Lookup Response Data 2034
Table 568: GET Security Identities Input Parameters 2035
Table 569: GET Security Identities Response Data 2036
Table 570: POST Security Identities Input Parameters 2039
Table 571: POST Security Identities Response Data 2039
Table 572: GET Security Containers {id} Roles Input Parameters 2040
Table 573: GET Security Containers {id} Roles Response Data 2040
Table 574: POST Security Containers {id} Roles Input Parameters 2041
Table 575: POST Security Containers {id} Roles Response Data 2041
Table 576: GET Security Audit Collections {id} Input Parameters 2042
Table 577: GET Security Audit Collections {id} Response Data 2043
Table 578: GET Security My Roles Response Data 2045
Table 579: Security Claims Endpoints 2046
Table 580: GET Security Claims Input Parameters 2047
Table 581: GET Security Claims Response Data 2048
Table 582: POST Security Claims Input Parameters 2050
Table 583: POST Security Claims Response Data 2052
Table 584: PUT Security Claims Input Parameters 2054
Table 585: PUT Security Claims Response Data 2055
Table 586: GET Security Claims{id} Input Parameters 2056
Table 587: GET Security Claims{id} Response Data 2057
Table 588: DELETE Security Claims{id} Input Parameters 2058
Table 589: GET Security Claims Roles Input Parameters 2060
Table 590: GET Security Claims Roles Response Data 2061
Table 591: Security Roles Permissions Endpoints 2062
Table 592: GET Security Roles {id} Permissions Input Parameters 2064
Table 593: GET Security Roles {id} Permissions Response Data 2064
Table 594: GET Security Roles {id} Global Permissions Input Parameters 2065
Table 595: GET Security Roles {id} Global Permissions Response Data 2065
Table 596: POST Security Roles {id}Global Permissions Input Parameters 2066
Table 597: POST Security Roles {id} Global Permissions Response Data 2066
Table 598: PUT Security Roles {id}Global Permissions Input Parameters 2068
Table 599: PUT Security Roles {id} Global Permissions Response Data 2068
Table 600: GET Security Roles {id} Permissions Containers Input Parameters 2069
Table 601: GET Security Roles {id} Permissions Containers Response Data 2069
Table 602: POST Security Roles {id} Permissions Containers Input Parameters 2071
Table 603: POST Security Roles {id} Permissions Containers Response Data 2071
Table 604: PUT Security Roles {id} Permissions Containers Input Parameters 2073
Table 605: PUT Security Roles {id} Permissions Containers Response Data 2073
Table 606: GET Security Roles {id} Permissions Collections Input Parameters 2074
Table 607: GET Security Roles {id} Permissions Collections Response Data 2074
Table 608: POST Security Roles {id} Permissions Collections Input Parameters 2076
Table 609: POST Security Roles {id} Permissions Collections Response Data 2076
Table 610: PUT Security Roles {id} Permissions Collections Input Parameters 2078
Table 611: PUT Security Roles {id} Permissions Collections Response Data 2078
Table 612: GET Security Roles {id} Permissions PAM Providers Input Parameters 2079
Table 613: GET Security Roles {id} Permissions PAM Providers Response Data 2079
Table 614: PUT Security Roles {id} Permissions PAM Providers Input Parameters 2080
Table 615: PUT Security Roles {id} Permissions PAM Providers Response Data 2080
Table 616: Security Roles Endpoints 2081
Table 617: DELETE Security Roles {id} Input Parameters 2082
Table 618: GET Security Roles {id} v2 Input Parameters 2083
Table 619: GET Security Roles {id} v2 Response Data 2084
Table 620: GET Security Roles {id} v1 Input Parameters 2086
Table 621: GET Security Roles {id} v1 Response Data 2087
Table 622: GET Security Roles v2 Input Parameters 2089
Table 623: GET Security Roles v2 Response Data 2090

Table 624: GET Security Roles v1 Input Parameters 2091
Table 625: GET Security Roles v1 Response Data 2092
Table 626: POST Security Roles v2 Input Parameters 2095
Table 627: POST Security Roles v2 Response Data 2098
Table 628: POST Security Roles v1 Input Parameters 2102
Table 629: POST Security Roles v1 Response Data 2104
Table 630: PUT Security Roles v2 Input Parameters 2107
Table 631: PUT Security Roles v2 Response Data 2110
Table 632: PUT Security Roles v1 Input Parameters 2114
Table 633: PUT Security Roles v1 Response Data 2116
Table 634: POST Security Roles {id} Copy Input Parameters 2118
Table 635: POST Security Roles {id} Copy Response Data 2119
Table 636: PUT Security Roles {id} Identities Input Parameters 2121
Table 637: PUT Security Roles {id} Identities Response Data 2121
Table 638: GET Security Roles {id} Identities Input Parameters 2122
Table 639: GET Security Roles {id} Identities Response Data 2122
Table 640: SSH Endpoints 2122
Table 641: SSH Keys Endpoints 2127
Table 642: DELETE SSH Keys Unmanaged {id} Input Parameters 2128
Table 643: GET SSH Keys Unmanaged {id} Input Parameters 2129
Table 644: GET SSH Keys Unmanaged {id} Response Data 2130
Table 645: GET SSH Keys My Key Input Parameters 2131
Table 646: GET SSH Keys My Key Response Data 2132
Table 647: POST SSH Keys My Key Input Parameters 2134
Table 648: POST SSH Keys My Key Response Data 2136
Table 649: PUT SSH Keys My Key Input Parameters 2138
Table 650: PUT SSH Keys My Key Response Data 2139
Table 651: DELETE SSH Keys Unmanaged Input Parameters 2140
Table 652: GET SSH Keys Unmanaged Input Parameters 2142
Table 653: GET SSH Keys Unmanaged Response Data 2143
Table 654: SSH Logon Endpoints 2144
Table 655: DELETE SSH Logons {id} Input Parameters 2145
Table 656: GET SSH Logons {id} Input Parameters 2145
Table 657: GET SSH Keys Unmanaged {id} Response Data 2146
Table 658: GET SSH Logons Input Parameters 2148
Table 659: GET SSH Logons Response Data 2149
Table 660: POST SSH Logons Input Parameters 2150
Table 661: POST SSH Logons Response Data 2151
Table 662: POST SSH Logons Access Input Parameters 2152
Table 663: POST SSH Logons Access Response Data 2153
Table 664: SSH Servers Endpoints 2153
Table 665: DELETE SSH Servers {id} Input Parameters 2154
Table 666: GET SSH Servers {id} Input Parameters 2155
Table 667: GET SSH Servers {id} Response Data 2156
Table 668: GET SSH Servers Access {id} Input Parameters 2160
Table 669: GET SSH Servers Access {id} Response Data 2161
Table 670: GET SSH Servers Input Parameters 2163
Table 671: GET SSH Servers Response Data 2164
Table 672: POST SSH Servers Input Parameters 2168
Table 673: POST SSH Servers Response Data 2169
Table 674: PUT SSH Servers Input Parameters 2173
Table 675: PUT SSH Servers Response Data 2174
Table 676: DELETE SSH Servers Access Input Parameters 2179
Table 677: DELETE SSH Servers Access Response Data 2180
Table 678: POST SSH Servers Access Input Parameters 2182
Table 679: POST SSH Servers Access Response Data 2183
Table 680: SSH Server Groups Endpoints 2184

Table 681: DELETE SSH Server Groups {id} Input Parameters 2185
Table 682: GET SSH Server Groups {id} Input Parameters 2186
Table 683: GET SSH Server Groups {id} Response Data 2187
Table 684: GET SSH Server Groups {name} Input Parameters 2190
Table 685: GET SSH Server Groups {name} Response Data 2191
Table 686: GET SSH Server Groups Access {id} Input Parameters 2195
Table 687: GET SSH Server Groups Access {id} Response Data 2196
Table 688: GET SSH Server Groups Input Parameters 2198
Table 689: GET SSH Server Groups Response Data 2199
Table 690: POST SSH Server Groups Input Parameters 2203
Table 691: POST SSH Server Groups Response Data 2207
Table 692: PUT SSH Server Groups Input Parameters 2211
Table 693: PUT SSH Server Groups Response Data 2215
Table 694: DELETE SSH Server Groups Access Input Parameters 2219
Table 695: DELETE SSH Server Groups Access {id} Response Data 2220
Table 696: POST SSH Server Groups Access Input Parameters 2222
Table 697: POST SSH Server Groups Access {id} Response Data 2223
Table 698: SSH Service Accounts Endpoints 2224
Table 699: DELETE SSH Service Accounts {id} Input Parameters 2226
Table 700: GET SSH Service Accounts {id} Input Parameters 2227
Table 701: GET SSH Service Accounts {id} Response Data 2228
Table 702: GET SSH Service Accounts Key {id} Input Parameters 2235
Table 703: GET SSH Service Accounts Key {id} Response Data 2237
Table 704: DELETE SSH Service Accounts Input Parameters 2239
Table 705: GET SSH Service Accounts Input Parameters 2241
Table 706: GET SSH Service Accounts Response Data 2242
Table 707: POST SSH Service Accounts Input Parameters 2249
Table 708: POST SSH Service Accounts Response Data 2252
Table 709: PUT SSH Service Accounts Input Parameters 2259
Table 710: PUT SSH Service Accounts Response Data 2260
Table 711: GET SSH Service Accounts Rotate {id} Input Parameters 2267
Table 712: GET SSH Service Accounts Rotate {id} Response Data 2269
Table 713: SSH Users Endpoints 2270
Table 714: DELETE SSH Users {id} Input Parameters 2270
Table 715: GET SSH Users {id} v2 Input Parameters 2271
Table 716: GET SSH Users {id} v2 Response Data 2272
Table 717: GET SSH Users {id} v1 Input Parameters 2274
Table 718: GET SSH Users {id} v1 Response Data 2275
Table 719: GET SSH Users v2 Input Parameters 2278
Table 720: GET SSH Users v2 Response Data 2281
Table 721: GET SSH Users v1 Input Parameters 2283
Table 722: GET SSH Users v1 Response Data 2286
Table 723: POST SSH Users Input Parameters 2288
Table 724: POST SSH Users Response Data 2288
Table 725: PUT SSH Users Input Parameters 2289
Table 726: POST SSH Users Response Data 2290
Table 727: POST SSH Users Access Input Parameters 2291
Table 728: POST SSH Users Access Response Data 2292
Table 729: SMTP Endpoints 2294
Table 730: GET SMTP Response Data 2295
Table 731: PUT SMTP Input Parameters 2297
Table 732: POST SMTP Test Response Data 2298
Table 733: POST SMTP Test Input Parameters 2300
Table 734: POST SMTP Test Response Data 2302
Table 735: SSL Endpoints 2303
Table 736: GET SSL Parts {id} Input Parameters 2305
Table 737: GET SSL Parts {id} Response Data 2306

Table 738: GET SSL Endpoints {id} Input Parameters 2308
Table 739: GET SSL Endpoints {id} Response Data 2308
Table 740: DELETE SSL Network Ranges {id} Input Parameters 2309
Table 741: GET SSL Network Ranges {id} Input Parameters 2309
Table 742: GET SSL Network Ranges {id} Response Data 2310
Table 743: GET SSL Networks {id} Input Parameters 2310
Table 744: GET SSL Networks {id} Response Data 2311
Table 745: GET SSL Input Parameters 2321
Table 746: GET SSL Response Data 2322
Table 747: GET SSL Networks Input Parameters 2323
Table 748: GET SSL Networks Response Data 2324
Table 749: POST SSL Networks Input Parameters 2334
Table 750: POST SSL Networks Response Data 2344
Table 751: PUT SSL Networks Input Parameters 2348
Table 752: PUT SSL Networks Response Data 2358
Table 753: GET SSL Endpoints {id} History Input Parameters 2361
Table 754: GET SSL Endpoints {id} History Response Data 2362
Table 755: GET SSL Networks {id} Parts Input Parameters 2367
Table 756: GET SSL Networks {id} Parts Response Data 2368
Table 757: POST SSL Network Ranges Input Parameters 2369
Table 758: PUT SSL Network Ranges {id} Input Parameters 2370
Table 759: PUT SSL Endpoints Review Status Input Parameters 2370
Table 760: PUT SSL Endpoints Monitor Status Input Parameters 2371
Table 761: PUT SSL Endpoints Review All Input Parameter 2372
Table 762: PUT SSL Endpoints Monitor All Input Parameter 2372
Table 763: POST SSL Networks {id} Scan Input Parameters 2373
Table 764: POST SSL Networks {id} Reset Input Parameters 2374
Table 765: POST SSL Network Ranges Validate Input Parameters 2374
Table 766: DELETE SSL Networks {id} Input Parameters 2375
Table 767: Status Endpoints 2375
Table 768: Templates Endpoints 2376
Table 769: GET Templates {id} Input Parameters 2377
Table 770: GET Templates {id} Response Data 2378
Table 771: GET Templates Settings Response Data 2396
Table 772: PUT Templates Settings Input Parameters 2404
Table 773: PUT Templates Settings Response Data 2413
Table 774: GET Templates Subject Parts Response Data 2421
Table 775: GET Templates Input Parameters 2423
Table 776: GET Templates Response Data 2425
Table 777: PUT Templates Input Parameters 2436
Table 778: PUT Templates Response Body 2453
Table 779: POST Templates Import Input Parameters 2470
Table 780: Workflow Certificates Endpoints 2470
Table 781: GET Workflow Certificates {id} Input Parameters 2472
Table 782: GET Workflow Certificates {id} Input Parameters 2473
Table 783: GET Workflow Certificates Denied Input Parameters 2476
Table 784: GET Workflow Certificates Denied Response Data 2477
Table 785: GET Workflow Certificates Pending Input Parameters 2479
Table 786: GET Workflow Certificates Pending Response Data 2480
Table 787: GET Workflow Certificates External Validation Input Parameters 2482
Table 788: GET Workflow Certificates External Validation Response Data 2483
Table 789: POST Workflow Certificates Deny Input Parameters 2484
Table 790: POST Workflow Certificates Deny Response Data 2485
Table 791: POST Workflow Certificates Approve Input Parameters 2486
Table 792: POST Workflow Certificates Approve Response Data 2487
Table 793: Workflow Definitions Endpoints 2488
Table 794: GET Workflow Definitions Steps {extensionName} Input Parameters 2489

Table 795: GET Workflow Definitions Steps {extensionName} Response Data 2490
Table 796: DELETE Workflow Definitions {definitionid} Input Parameters 2494
Table 797: GET Workflow Definitions {definitionid} Input Parameters 2495
Table 798: GET Workflow Definitions {definitionsid} Response Data 2496
Table 799: PUT Workflow Definitions {definitionid} Input Parameters 2519
Table 800: PUT Workflow Definitions {definitionid} Response Data 2520
Table 801: GET Workflow Definitions Input Parameters 2543
Table 802: GET Workflow Definitions Response Data 2544
Table 803: POST Workflow Definitions Input Parameters 2546
Table 804: POST Workflow Definitions Response Data 2548
Table 805: GET Workflow Definitions Steps Input Parameters 2571
Table 806: GET Workflow Definitions Steps Response Data 2572
Table 807: GET Workflow Definitions Types Input Parameters 2577
Table 808: GET Workflow Definitions Types Response Data 2578
Table 809: PUT Workflow Definitions {definitionid} Steps Input Parameters 2580
Table 810: PUT Workflow Definitions {definitionid} Steps Response Data 2582
Table 811: POST Workflow Definitions {definitionid} Publish Input Parameters 2605
Table 812: POST Workflow Definitions {definitionid} Publish Response Data 2606
Table 813: Workflow Instances Endpoints 2629
Table 814: DELETE Workflow Instances {instanceid} Input Parameters 2630
Table 815: GET Workflow Instances {instanceId} Input Parameters 2630
Table 816: GET Workflow Instances {instanceId} Response Data 2631
Table 817: GET Workflow Instances Input Parameters 2656
Table 818: GET Workflow Instances Response Data 2657
Table 819: GET Workflow Instances My Input Parameters 2661
Table 820: GET Workflow Instances My Response Data 2662
Table 821: GET Workflow Instances AssignedToMe Input Parameters 2666
Table 822: GET Workflow Instances AssignedToMe Response Data 2667
Table 823: POST Workflow Instances {instanceid} Stop Input Parameters 2670
Table 824: POST Workflow Instances {instanceid} Signals Input Parameters 2672
Table 825: POST Workflow Instances {instanceid} Restart Input Parameters 2673
Table 826: API Change Log v9.0 2675
Table 827: API Change Log v9.1 2677
Table 828: API Change Log v9.2 2678
Table 829: API Change Log v9.3 2678
Table 830: API Change Log v9.4 2678
Table 831: API Change Log v9.5 2678
Table 832: API Change Log v9.7 2679
Table 833: API Change Log v9.9 2679
Table 834: API Change Log v10.0 2681
Table 835: API Change Log v10.1 2686
Table 836: API Change Log v10.2 2686
Table 837: API Change Log v10.3.1 2687
Table 838: API Change Log v10.4 2687
Table 839: API Change Log v10.4.3 2688
Table 840: API Change Log v10.4.5 2688
Table 841: API Change Log v10.4.6 2689
Table 842: API Change Log v11.0 2690
Table 843: System Requirements 2702
Table 844: Keyfactor Identity Provider Container Parameters 2714
Table 845: Typical Service Accounts 2762
Table 846: Protocols Keyfactor Command Uses for Communication 2766
Table 847: .NET Framework Release Values 2768
Table 848: Available components for Keyfactor. 2782
Table 849: Identity Provider Parameters 2791
Table 850: Features Required for Each Server Role 2816
Table 851: Input Parameters XML File Fields 2818

Table 852: ConfigurationWizardConsole.exe Options 2821
Table 853: Microsoft CA Permission Matrix 2834
Table 854: Linux Container Parameters 2932
Table 855: Remote CA Configuration Parameters 2939
Table 856: API Change Log 3087
Table 857: API Change Log 3105
Table 858: API Change Log 3113
Table 859: API Change Log 3115
Table 860: API Change Log 3117
Table 861: API Change Log 3123
Table 862: API Change Log 3124
Table 863: API Change Log 3128
Table 864: API Change Log 3130
Table 865: Keyfactor Universal Orchestrator vs Windows Orchestrator Capabilities 3142
Table 866: API Change Log 3143
Table 867: API Change Log 3147
Table 868: API Change Log 3150
Table 869: API Change Log 3152
Table 870: API Change Log 3155
Table 871: API Change Log 3156
Table 872: API Change Log 3160
Table 873: API Change Log 3163
Table 874: Compatibility Matrix for Keyfactor Command v11 3164
Table 875: Compatibility Matrix for Keyfactor Command v10 3167
Table 876: Compatibility Matrix for Keyfactor Command v9 3170

List of Figures

Figure 1: Management Portal Menu 2
Figure 2: Using the Management Portal Grids 4
Figure 3: Under Construction Icon 5
Figure 4: Confirmation Message 5
Figure 5: Dashboard Risk Header 6
Figure 6: Click the Dashboard Add Panel Button 8
Figure 7: Add Panels to the Dashboard 9
Figure 8: Dashboard Panel Settings 10
Figure 9: Type in a New Name for the Panel 10
Figure 10: Dashboard Panel Settings 10
Figure 11: Dashboard CA Snapshot 12
Figure 12: Dashboard Certificate Collections 13
Figure 13: Dashboard Certificates by Signing Algorithm 14
Figure 14: Dashboard SSH Keys per Type 15
Figure 15: Dashboard Recent Certificate Store Jobs 16
Figure 16: Dashboard Revocation Monitoring Status 16
Figure 17: Dashboard SSL Endpoints 18
Figure 18: Dashboard SSL Orchestrator Job Status 19
Figure 19: Certificate Details: Content Tab 20
Figure 20: Certificate Details: Metadata Tab 22
Figure 21: Email Type Metadata 23
Figure 22: Certificate Details: Status Tab 24
Figure 23: Certificate Details: Validation Tab 27
Figure 24: Location Details 31
Figure 25: Total Certificate Store Location Details 31
Figure 26: Certificate Operation: Certificate History Tab 33
Figure 27: Certificate Operation: Certificate History Detail 34
Figure 28: Certificate Search 39
Figure 29: Save Certificate Collection 43
Figure 30: Select Certificate Store Locations Dialog 47
Figure 31: Add Certificate—Install into Certificate Locations 49
Figure 32: Alias Required Alert on Save 49
Figure 33: Certificate Store Type Configuration: Basic Tab 50
Figure 34: Example: Certificate Location Details for a JKS Location 52
Figure 35: Certificate Store Type Configuration: Advanced Tab 53
Figure 36: Certificate Store Type Configuration: Entry Parameters Tab 54
Figure 37: Certificate Operation: Download Certificate with Private Key 57
Figure 38: Certificate Operation: Password for Certificate with Private Key 58
Figure 39: Download a Certificate with Custom Password 59
Figure 40: Certificate Operation: Download Certificate without Private Key 60
Figure 41: Certificate Operation: Edit All 63
Figure 42: Certificate Operation: Edit All Alerts 64
Figure 43: IIS Setting for 1+ Million Records - Certificate Operation: Edit All 65
Figure 44: Certificate Operation: CSV Download 66
Figure 45: Certificate Operation: Identity Audit 67
Figure 46: Certificate Operation: Select Stores for Remove from Certificate Store 68
Figure 47:Remove from Cert Store Save Page 69
Figure 48: Certificate Operation: Renew/Reissue with the Continue Option 70
Figure 49: Certificate Operation: Revoke 72
Figure 50: Certificate Operation: Revoke All 74
Figure 51: Add Certificate Password for PFX/p12 75
Figure 52: Add Certificate Information 76
Figure 53: Add Certificate Metadata 76

Figure 54: Select Certificate Store Locations Dialog 78
Figure 55: Add Certificate—Install into Certificate Locations 79
Figure 56: Alias Required Alert on Save 79
Figure 57: Certificate Store Type Configuration: Basic Tab 80
Figure 58: Example: Certificate Location Details for a JKS Location 82
Figure 59: Certificate Store Type Configuration: Advanced Tab 83
Figure 60: Certificate Store Type Configuration: Entry Parameters Tab 84
Figure 61: Certificate Collection Manager 86
Figure 62: View Collection 87
Figure 63: Collection with Query Modification 88
Figure 64: Report Drill Down: Certificates by Key Strength Report 93
Figure 65: Report Drill Down: Certificate Search Results 93
Figure 66: Certificate Count by Template: Issued Certificates 95
Figure 67: Certificate Count by User by Template 96
Figure 68: Certificate Count Grouped by Single Metadata Field 98
Figure 69: Certificate Issuance Trends with Metadata: Requesters 99
Figure 70: Certificate Issuance Trends with Metadata: Metadata Table and Chart 99
Figure 71: Certificates by Key Strength 100
Figure 72: Certificates by Revoker 101
Figure 73: Certificates by Type and Java Keystore 102
Figure 74: Certificates Found at TLS/SSL Endpoints 103
Figure 75: Certificate Expiration Report: Certificates Expiring within One Week 105
Figure 76: Issued Certificates per CA 112
Figure 77: Example Pie Chart from Monthly Executive Report 114
Figure 78: PKI Status for Collection Summary 115
Figure 79: PKI Status for Collection Lifetime Remaining 116
Figure 80: PKI Status for Collection Top Issuers 118
Figure 81: PKI Status for Certificates issued in previous 10 weeks 118
Figure 82: PKI Status for Certificates issued in previous 12 months 119
Figure 83: Example Portion of the Statistical Report 125
Figure 84: Report Manager Grid 127
Figure 85: Edit a Report in Report Manager Details Tab 129
Figure 86: Edit a Report in Report Manager Parameters Tab 131
Figure 87: Report Manager Parameters Tab: Parameter Details 131
Figure 88: Edit a Report in Report Manager Schedule Tab 132
Figure 89: Edit a Report in Report Manager Schedule Tab - Add/Edit page 133
Figure 90: CSR Enrollment: CSR Content 137
Figure 91: CSR Enrollment: CSR Names 138
Figure 92: Select a Certificate Template 138
Figure 93: CSR Enrollment for Stand-Alone CA 139
Figure 94: CSR Enrollment SAN options 139
Figure 95: Populate Enrollment Fields 140
Figure 96: Populate Metadata Fields 141
Figure 97: Select a Certificate Format 141
Figure 98: CSR Enrollment Completed Successfully—Awaiting Workflow Approval(s) 142
Figure 99: CSR Enrollment Completed Successfully—Pending Status 142
Figure 100: CSR Generation 144
Figure 101: CSR Generation SAN Options 145
Figure 102: CSR Generation Success 145
Figure 103: Pending CSRs 146
Figure 104: Select a Certificate Template 147
Figure 105: PFX Enrollment for Stand-Alone CA 148
Figure 106: PFX Enrollment for ECC Template Displaying Elliptic Curve 148
Figure 107: PFX Enrollment 149
Figure 108: PFX Enrollment: SAN Options 150
Figure 109: Populate Enrollment Fields 151
Figure 110: Populate Metadata Fields 151

Figure 111: Set a Custom Password 152
Figure 112: Delivery Format PFX Enrollment 153
Figure 113: Select Certificate Store Locations Dialog 154
Figure 114: PFX Enrollment: Certificate Delivery Format 156
Figure 115: Alias Required System Alert on Enrolling 156
Figure 116: Certificate Store Type Configuration: Basic Tab 157
Figure 117: Example: Certificate Location Details for a JKS Location 159
Figure 118: Certificate Store Type Configuration: Advanced Tab 160
Figure 119: Certificate Store Type Configuration: Entry Parameters Tab 161
Figure 120: PFX Enrollment Completed Successfully—Network Password Used 162
Figure 121: PFX Enrollment Completed Successfully—Awaiting Workflow Approval(s) 162
Figure 122: PFX Enrollment Completed Successfully—Pending Status 163
Figure 123: Certificate Requests Grid 165
Figure 124: Certificate Request Details 165
Figure 125: Certificate Template Requiring Manager Approval 166
Figure 126: Create a New Expiration Alert 168
Figure 127: Expiration Alerts Recipients 171
Figure 128: Expiration Alert Schedule 172
Figure 129: Expiration Alert Test 174
Figure 130: Certificate Template Requiring Manager Approval 179
Figure 131: Create a New Pending Request Alert 181
Figure 132: Pending Request Alerts Recipients 183
Figure 133: Pending Request Alert Schedule 184
Figure 134: Pending Alert Test 185
Figure 135: Create a New Issued Certificate Alert 190
Figure 136: Issued Certificate Alerts Recipients 193
Figure 137: Issued Alert Schedule 194
Figure 138: Create a New Denied Certificate Request Alert 199
Figure 139: Denied Certificate Request Alerts Recipients 201
Figure 140: Key Rotation Alerts Recipients 204
Figure 141: Substitutable Special Text for Key Rotation Alerts 205
Figure 142: Key Rotation Alert Schedule 207
Figure 143: Key Rotation Alert Viewer 210
Figure 144: Revocation Monitoring Grid 211
Figure 145: CRL Monitoring Details 213
Figure 146: OCSP Monitoring Details 217
Figure 147: Test Revocation Monitoring 218
Figure 148: Revocation Monitoring Event Log Messages 218
Figure 149: Use PowerShell Expiration Event Handler 223
Figure 150: Expiration Alert with PowerShell Event Handler 223
Figure 151: PowerShell Event Handler with Multiple Parameters 224
Figure 152: Expiration Alert with Event Logging Event Handler 225
Figure 153: Expiration Alert with Logging Event Handler 226
Figure 154: Expiration Alert Event Log 227
Figure 155: Use Renewal Event Handler on Expiration Alert 228
Figure 156: Expiration Alert with URL Event Handler 229
Figure 157: Workflow Definitions 234
Figure 158: Using the Workflow Workspace 236
Figure 159: Edit PowerShell Window 236
Figure 160: Edit Content Window 237
Figure 161: Create a New Workflow Definition 238
Figure 162: Click Plus to Add a New Workflow Definition Step 239
Figure 163: Select a Workflow Definition Step 240
Figure 164: Display Name is Step Name Title 241
Figure 165: Signals Configuration for a Requires Approval Workflow Definition Step 242
Figure 166: Export Workflow Definition 245
Figure 167: Browse to Locate a Workflow Definition to Import 247

Figure 168: Workflow Definition Versions: View Current Version 248
Figure 169: Workflow Definition Versions: View Previous Version 248
Figure 170: Tokens are Highlighted 249
Figure 171: Conditions Example: Add Parameters 250
Figure 172: Conditions Example: Add Conditions for Require Approval Step 251
Figure 173: Tokens are Highlighted 252
Figure 174: Requester Lookup Example: Add Parameters 255
Figure 175: Requester Lookup Example: Add Headers for REST Request 256
Figure 176: Tokens are Highlighted 258
Figure 177: Metadata Update Example: Add Parameters 262
Figure 178: Metadata Update Example: Add Headers for REST Request 263
Figure 179: Metadata Update Example: Results 264
Figure 180: Tokens are Highlighted 264
Figure 181: Configuration Parameters for a Require Approval Workflow Definition Step 266
Figure 182: Use Custom PowerShell with Embedded REST Request: Add Parameters 270
Figure 183: Tokens are Highlighted 273
Figure 184: Step Configuration for an Email Workflow Definition Step 275
Figure 185: Add Parameters for PowerShell 276
Figure 186: Configuration Parameters for a Set Variable Data Workflow Definition Step 277
Figure 187: Revocation Comment Update Example: Add Parameters 278
Figure 188: Revocation Comment Update Example: Results 279
Figure 189: Additional Attribute Update Example: Add Parameters 280
Figure 190: Add Parameters for PowerShell 282
Figure 191: Step Configuration for a Custom PowerShell Workflow Definition Step 283
Figure 192: Update SANs Example: Add Parameters 286
Figure 193: Approval Comment Update Example: Add Parameters 288
Figure 194: Approval Comment Update Example: Results 288
Figure 195: Update Certificate Request Subject\SANs for Microsoft CAs Workflow Definition Step 291
Figure 196: Update SANs and Subject Example: Add Parameters 292
Figure 197: Simple Workflow Definitions Search 304
Figure 198: PFX Enrollment Complete for a Template Requiring Approval via Workflow 305
Figure 199: View Workflow Instance for a PFX Enrollment 306
Figure 200: Workflow Instances 307
Figure 201: Workflow Instance Review 310
Figure 202: View a Workflow Instance 317
Figure 203: View an Audit Log Entry for a Restarted Workflow Instance 320
Figure 204: Simple Workflow Instance Search 323
Figure 205: Workflows Assigned to Mary 325
Figure 206: Workflow Instance Review 327
Figure 207: Approve or Deny a Workflow Instance 332
Figure 208: Simple Workflows Assigned to Me Search 335
Figure 209: Workflow Instance Review 338
Figure 210: View Details for the Workflow Instance 345
Figure 211: Simple Workflows Created by Me Search 347
Figure 212: Import Certificate Authorities 353
Figure 213: Save Certificate Authority 354
Figure 214: Enforce unique DN Setting on the EJBCA CA 358
Figure 215: EJBCA Certificate Profile Validity Offset Greater than 15 Minutes 361
Figure 216: EJBCA Certificate Profile Backdated Revocation 361
Figure 217: Certificate Authority Basic Tab for a Microsoft CA 363
Figure 218: Certificate Authority Basic Tab for an AnyGateway REST CA 365
Figure 219: Certificate Authority Advanced Tab for Microsoft CA 367
Figure 220: Certificate Authority Authentication Methods Tab for a DCOM-Microsoft CA 375
Figure 221: Certificate Authority Authentication Methods Tab for an HTTPS-EJBCA CA 375
Figure 222: Certificate Authority Standalone Tab 377
Figure 223: Certificate Authority Monitoring Recipients 379
Figure 224: Certificate Templates 380

Figure 225: Configure System-Wide Enrollment Regular Expressions 384
Figure 226: Configure System-Wide Enrollment Defaults 385
Figure 227: Configure System-Wide Policies 387
Figure 228: Microsoft Issuance Requirements on a Template for Manager Approval 389
Figure 229: Certificate Template: Details Tab for a Microsoft Template 390
Figure 230: Configure Template: Enrollment Fields Tab 391
Figure 231: Certificate Template: Authorization Methods Tab 393
Figure 232: Certificate Template: Metadata Tab 394
Figure 233: Certificate Template: Enrollment RegExesTab 396
Figure 234: Certificate Template: Template Regular Expression Error on Enrollment 397
Figure 235: Certificate Template: Enrollment Defaults Tab 398
Figure 236: Certificate Template: Policies Tab 402
Figure 237: PFX Enrollment Regular Expression Validation Error 405
Figure 238: Simple Certificate Store Search 412
Figure 239: Add a Remote JKS Certificate Store 415
Figure 240: Set Password Dialog 417
Figure 241: View Details for a Certificate Store 425
Figure 242: Enter a Information for Reenrollment 427
Figure 243: View Inventoried Certificates for a Certificate Store 430
Figure 244: Schedule Inventory for a Certificate Store Location 431
Figure 245: Certificate Store Container Search 434
Figure 246: Certificate Store Containers 435
Figure 247: Define a Certificate Store Container 436
Figure 248: Schedule Java Keystore Discover Job for Remote File Extension 438
Figure 249: Schedule F5 SSL Discover Job for F5 Extension 439
Figure 250: Discovered Certificate Stores 443
Figure 251: Java Keystore Set Password 444
Figure 252: F5 SSL Profiles Set Password 446
Figure 253: Manage a Discovered Java Certificate Store 448
Figure 254: F5 SSL Profiles Set Password 449
Figure 255: Manage a Discovered F5 SSL Profile Certificate 451
Figure 256: SSL Network Discovery 456
Figure 257: Define a New Network—Basic Tab 458
Figure 258: Define a New Network—Advanced Tab 460
Figure 259: Define a New Network—Network Ranges Tab 462
Figure 260: Define a New Network—Quiet Hours Tab 464
Figure 261: SSL Network Scan Details Page 465
Figure 262: SSL Network Scan Detail Segment Details 466
Figure 263: SSL Network ScanNow 467
Figure 264: SSL Orchestrator Pools 470
Figure 265: Add an Orchestrator Pool 471
Figure 266: SSL Discovery Results 473
Figure 267: SSL Discovery and Monitoring Result Details 478
Figure 268: SSL Discovery Email 480
Figure 269: SSL Monitoring Email 480
Figure 270: Orchestrator Auto-Registration Settings Page 493
Figure 271: Orchestrator Auto-Registration Edit 494
Figure 272: Orchestrator Auto-Registration Flow 496
Figure 273: View Details for an Orchestrator 500
Figure 274: Generate a Blueprint from an Existing Orchestrator 501
Figure 275: Apply a Blueprint from a New Orchestrator 501
Figure 276: Reset an Orchestrator 502
Figure 277: Request Renewal for an Orchestrator 503
Figure 278: View Active Jobs for an Orchestrator 503
Figure 279: View Job History for an Orchestrator 504
Figure 280: View Certificate Stores for an Orchestrator 504
Figure 281: Sample Native Agent Fetch Log Results 506

Figure 282: Modify IIS Settings for Keyfactor Universal Orchestrator Custom Jobs: maxAllowedContentLength 507
Figure 283: Orchestrator Job Status Scheduled Jobs 512
Figure 284: Orchestrator Job History 517
Figure 285: Orchestrator Blueprints 522
Figure 286: Orchestrator Blueprint Details: Certificate Stores Tab 523
Figure 287: Orchestrator Blueprint Details: Scheduled Jobs Tab 524
Figure 288: Mac Auto-Enrollment Configuration 525
Figure 289: SSH Key Discovery Flow 526
Figure 290: SSH User Key Management Flow 526
Figure 291: Add SSH Server Group for Discovery 529
Figure 292: Add SSH Server for Discovery 529
Figure 293: Use PuTTY Key Generator to Convert Zed's Private Key 532
Figure 294: Create Logons and Mappings for Zed 532
Figure 295: Configure PuTTY to Use Zed's Private Key 533
Figure 296: Key Information for an SSH User Key 535
Figure 297: Generate an SSH Key Pair 536
Figure 298: Rotate an SSH Key Pair 539
Figure 299: Add a Password to Encrypt the Downloaded Private Key 541
Figure 300: Edit SSH User Key Information 542
Figure 301: Acquire a New Service Account Key 544
Figure 302: Map Service Account Public Key to Logon 545
Figure 303: Add a Service Account Key 546
Figure 304: Edit SSH Service Account Key Information 549
Figure 305: Rotate an SSH Key Pair 551
Figure 306: Download a Service Account Private Key 553
Figure 307: View Basic Tab of an Unmanaged SSH Key 557
Figure 308: View Logon Tab of an Unmanaged SSH Key 557
Figure 309: SSH Server Groups Grid 561
Figure 310: Add a Server Group 562
Figure 311: Edit Access for an SSH Server Group 564
Figure 312: Edit Access for an SSH Server 565
Figure 313: Linux Logon to Keyfactor User Mappings for Anne, Betty, Chuck and Dave 567
Figure 314: Server Group Access Editing Example 568
Figure 315: Concept: Add Linux Logon for Chuck on Server C 569
Figure 316: Server Group Access: Add Linux Logon for Chuck on Server C 570
Figure 317: Add Logon to User Mapping for Betty 571
Figure 318: Remove Logon to User Mapping for Betty 572
Figure 319: Add Individual Logon to User Mappings for Dave 573
Figure 320: View Server Group Logon to User Mappings for Dave 574
Figure 321: View Members of an SSH Server Group 574
Figure 322: SSH Servers Grid 577
Figure 323: Add an SSH Server 578
Figure 324: Edit Access for an SSH Server 580
Figure 325: Edit Access for an SSH Server 582
Figure 326: Linux Logons Grid 585
Figure 327: Add a Linux Logon—Basic Tab 586
Figure 328: Add a Linux Logon—Access Management Tab 587
Figure 329: Edit Access for a Linux Logon 589
Figure 330: Creating Linux Logon to Keyfactor User Mappings Using Active Directory Groups Key Value 590
Figure 331: SSH Users Grid 593
Figure 332: Edit Access for a Keyfactor User 594
Figure 333: System Settings Icon 600
Figure 334: Console Application Settings: General 602
Figure 335: Console Application Settings: Monitoring 603
Figure 336: EJBCA Certificate Profile Validity Offset Greater than 15 Minutes 604
Figure 337: Audit Log Application Settings 608
Figure 338: Enrollment Application Settings 609

Figure 339: Agents Application Settings 614
Figure 340: API Application Settings 619
Figure 341: SSH Settings 620
Figure 342: Workflow Settings 621
Figure 343: Security Roles 623
Figure 344: Security Claims 624
Figure 345: Certificate Collection System-wide Permissions 628
Figure 346: Certificate Collection per Collection Permissions 629
Figure 347: Certificate Store Management—Global Permissions 630
Figure 348: Certificate Store Management - Container Permissions 630
Figure 349: Global PAM Permissions 631
Figure 350: PAM Provider Permissions 632
Figure 351: Grant Global Permissions to a Security Role 684
Figure 352: Grant Collection Permissions to a Security Role 685
Figure 353: Grant Container Permissions to a Security Role 686
Figure 354: Add a Security Claim for an OAuth Identity Provider Role 688
Figure 355: Associate Existing Claims with a Security Role 689
Figure 356: Add a Security Claim for an OAuth Identity Provider Role 694
Figure 357: View Global Permissions for a Security Claim 696
Figure 358: Collection Permissions for a Security Claim 697
Figure 359: Container Permissions for a Security Claim 697
Figure 360: View Roles associated with a Security Claim 698
Figure 361: Certificate Store Types 701
Figure 362: Add New Certificate Store Type: Basic Tab 702
Figure 363: Add New Certificate Store Type: Advanced Tab 704
Figure 364: Add New Certificate Store Type: Custom Fields Tab 706
Figure 365: Add New Certificate Store Type: Entry Parameters Tab 708
Figure 366: Certificate Metadata 711
Figure 367: Create or Edit Certificate Metadata Field 712
Figure 368: Metadata Hints in a Certificate Details Dialog 713
Figure 369: Metadata Display Order 715
Figure 370: Audit Log 717
Figure 371: Audit Log Search Selections for Template Property Field Search 720
Figure 372: Audit Log Details: Entry Metadata Section 724
Figure 373: Audit Log Details: Related Entries Section 724
Figure 374: Audit Log Details: Single Column Audit Details Pane 725
Figure 375: Audit Log Details: Two Column Audit Details Pane 725
Figure 376: Audit Log Details Dialog 726
Figure 377: Audit Log Record is Valid 727
Figure 378: Audit Log Details Showing Valid Status 727
Figure 379: Audit Log Details Showing Invalid Status 727
Figure 380: Management Portal Access Denied Message 731
Figure 381: Audit Log Authorization Failure Messages 731
Figure 382: Authorization Failure Audit Log Detail 733
Figure 383: Audit Logs for Certificates 734
Figure 384: Audit Log Details for Security 735
Figure 385: Audit Logs for SSH Management 736
Figure 386: Automated Entries Created by the System in the Audit Log 736
Figure 387: Audit Log Entries for Workflow 737
Figure 388: Security Role Showing Auditing Permissions Setting 739
Figure 389: Event Handler Registration Grid 741
Figure 390: Event Handler Registration 741
Figure 391: Event Handler Registration Editor 742
Figure 392: View Packages as Part of a List 744
Figure 393: View Packages on Individual Pages 744
Figure 394: Find the Latest Version of the Package 744
Figure 395: Download the Package Zip File 745

Figure 396: Remote PAM Provider 751
Figure 397: Create CyberArk Provider 752
Figure 398: Create Delinea PAM Provider 753
Figure 399: Create HashiCorp PAM Provider 754
Figure 400: Details for an Identity Provider 755
Figure 401: Edit Parameters for an Identity Provider 756
Figure 402: SMTP Configuration 766
Figure 403: Send an SMTP Test Message 767
Figure 404: Component Installations 767
Figure 405: Keyfactor Command License 768
Figure 406: Upload a New Keyfactor Command License 769
Figure 407: Save a New Keyfactor Command License 769
Figure 408: Sample WebAgentServices Appsettings.json File 775
Figure 409: Sample ClaimsProxy Appsettings.json File 777
Figure 410: Sample KeyfactorAPI Appsettings.json File 778
Figure 411: Appsettings.json File for TimerService Settings 780
Figure 412: Sample WebConsole Services Appsettings.json File 787
Figure 413: Sample Appsettings.json File for SQL Retry Settings 788
Figure 414: Switch the Keyfactor Command Service to Run as the CMSTimerService.dll 789
Figure 415: Active Directory Account Properties 792
Figure 416: Management Portal Errors and Warnings 794
Figure 417: Nlog Configuration for Windows Event Logging 796
Figure 418: Nlog_KeyfactorAPI.config 801
Figure 419: License Expiration Event Log 818
Figure 420: Upload a New Keyfactor Command License 819
Figure 421: IIS Authentication for Virtual Directories with an Identity Provider Other Than Active Directory 824
Figure 422: IIS Authentication for Virtual Directories with Active Directory 825
Figure 423: EJBCA Certificate Profile Custom Certificate Extensions 828
Figure 424: Certificate Validation Fails for Full Chain and CRL Online 829
Figure 425: Modify IIS Settings for SSL Scanning: maxAllowedContentLength 830
Figure 426: Modify IIS Settings for SSL Scanning:uploadReadAheadSize 831
Figure 427: Modify IIS Settings for SSL Scanning: maxRequestLength 832
Figure 428: Disable Loopback Checking: DisableStrictNameChecking 833
Figure 429: Disable Loopback Checking: BackConnectionHostNames 834
Figure 430: Client Secret for Keyfactor API in Keyfactor Identity Provider 845
Figure 431: Access Token for the Keyfactor API Reference and Utility 846
Figure 432: Keyfactor API Reference and Utility Authorize Options 847
Figure 433: Enter Access Token in the Keyfactor API Reference and Utility 848
Figure 434: Successful Authorization in the Keyfactor API Reference and Utility 848
Figure 435: Select a Version in the Keyfactor API Reference and Utility 852
Figure 436: Documentation in the Help Dropdown 853
Figure 437: Microsoft Issuance Requirements on a Template for Manager Approval 2392
Figure 438: Microsoft Issuance Requirements on a Template for Manager Approval 2433
Figure 439: Microsoft Issuance Requirements on a Template for Manager Approval 2451
Figure 440: Microsoft Issuance Requirements on a Template for Manager Approval 2467
Figure 441: Keyfactor Command Logical Architecture Diagram 2696
Figure 442: Keyfactor Command Physical Architecture Diagram 2700
Figure 443: Simple Keyfactor Command Solution Design 2701
Figure 444: Select the Download Hosting Bundle Option Under Run Server Apps 2703
Figure 445: Export the SQL Server Certificate as a P7B 2707
Figure 446: Add a SQL Authentication Login 2709
Figure 447: Add a SQL Database 2709
Figure 448: Select a Realm in the Keyfactor Identity Provider Administration Console 2716
Figure 449: Select Command-OIDC-Client in the Keyfactor Identity Provider Administration Console 2717
Figure 450: Regenerate the Keyfactor Identity Provider Secret 2717
Figure 451: Copy the Keyfactor Identity Provider Secret 2718
Figure 452: Set the Client Access Settings 2719

Figure 453: OpenID Endpoint Configuration Link 2720
Figure 454: OpenID Endpoint Configuration Settings 2721
Figure 455: SSO Session Values 2721
Figure 456: Access Token Lifespan 2722
Figure 457: Add a Keyfactor Identity Provider User 2723
Figure 458: Locate the Keyfactor Identity Provider User's ID 2723
Figure 459: Set a Password for the Keyfactor Identity Provider User 2724
Figure 460: Select a Realm in the Keyfactor Identity Provider Administration Console 2725
Figure 461: Add a Keyfactor Identity Provider Role 2725
Figure 462: Add a Keyfactor Identity Provider Group 2726
Figure 463: Assign a Role to a Group in Keyfactor Identity Provider 2726
Figure 464: Select a Realm in the Keyfactor Identity Provider Administration Console 2727
Figure 465: Add a Keyfactor Identity Provider User 2728
Figure 466: Set a Password for the Keyfactor Identity Provider User 2729
Figure 467: Assign a Role to a Keyfactor Identity Provider User 2730
Figure 468: Select a Realm in the Keyfactor Identity Provider Administration Console 2731
Figure 469: Add a Keyfactor Identity Provider Service Account (Client): General 2731
Figure 470: Add a Keyfactor Identity Provider Service Account (Client): Capabilities 2732
Figure 471: Copy the Keyfactor Identity Provider Service Account (Client) Secret 2732
Figure 472: Login Page with Choice of Federated Identity Provider 2733
Figure 473: Federated Identity Provider Login Flow 2734
Figure 474: Client ID and Secret in Okta OIDC Application 2735
Figure 475: Redirect URIs for the Okta OIDC Application 2736
Figure 476: Create an Authorization Server Role Claim 2737
Figure 477: Select a Realm in the Keyfactor Identity Provider Administration Console 2738
Figure 478: Give the Keyfactor Identity Provider Identity Provider an Alias 2738
Figure 479: Enter the Okta Discovery Endpoint in the Keyfactor Identity Provider Identity Provider 2739
Figure 480: Enable PKCE in the Keyfactor Identity Provider Identity Provider 2739
Figure 481: Add Okta Client ID and Secret in the Keyfactor Identity Provider Identity Provider 2739
Figure 482: Deliver the Okta openid and profile to the Keyfactor Identity Provider Identity Provider 2740
Figure 483: Map the Okta preferred_username to the Keyfactor Identity Provider Identity Provider Username 2741
Figure 484: Map the Okta Roles to the Keyfactor Identity Provider Identity Provider Roles 2742
Figure 485: SQL Server Configuration Manager View Active SSL Certificate 2747
Figure 486: Registry View Active SSL Certificate 2748
Figure 487: View SQL Server Services 2748
Figure 488: SQL Server SSL Certificate Details 2749
Figure 489: Grant Private Key Permissions for SQL Server 2750
Figure 490: Default SQL Connection Strings 2751
Figure 491: SQL Connection Strings with Encrypt Channel Disabled 2751
Figure 492: SQL Connection Strings with MultiSubnetFailover Option Enabled 2752
Figure 493: Certificate Template with Key Encipherment Key Usage 2753
Figure 494: Local Security Policy 2758
Figure 495: Install CA Chain Certificates on the Keyfactor Command Server 2764
Figure 496: Use Get-WindowsFeature to Determine if All Required Roles and Features are Installed 2770
Figure 497: Web Server Role 2770
Figure 498: .NET 4.7 Feature 2771
Figure 499: Role Services Page One 2772
Figure 500: Role Services Page Two 2773
Figure 501: Active Directory Module for Windows PowerShell 2773
Figure 502: Install: Begin Setup Wizard 2781
Figure 503: Install: Select Components 2782
Figure 504: Windows Authentication 2783
Figure 505: SQL Authentication 2783
Figure 506: Configure: Backup Database Master Key 2785
Figure 507: Configure: Upload License 2785
Figure 508: Configure: Open Data File 2786
Figure 509: Configure: Application Pools 2787

Figure 510: Configure: Identity Providers—OAuth Claims Proxy Section 2789
Figure 511: Configure: Identity Providers—OAuth Identity Provider Section 2791
Figure 512: Configure: Encryption Warning 2800
Figure 513: Configure: Database 2801
Figure 514: Configure: Service 2802
Figure 515: Configure: Email 2803
Figure 516: Configure: Keyfactor Portal 2805
Figure 517: Configure: Administrative Users for Active Directory 2806
Figure 518: Configure: Administrative Users for OAuth 2807
Figure 519: Configure: Dashboard and Reports 2808
Figure 520: Configure: Orchestrators with Standard Authentication 2809
Figure 521: Configure: Orchestrators with Client Certificate Authentication 2810
Figure 522: Configure: API 2811
Figure 523: Configure: Audit 2812
Figure 524: Configure: Configuration Warnings 2812
Figure 525: Configure: Save Configuration as a File 2813
Figure 526: Configure: Configuration Operations 2814
Figure 527: Configure: Configuration Complete 2814
Figure 528: Configure Local Intranet Zone in Internet Properties 2823
Figure 529: Configure Kerberos Constrained Delegation on the Keyfactor Command Machine Account 2826
Figure 530: Add HOST and rpcss Service Types for Kerberos Constrained Delegation 2827
Figure 531: Configure Kerberos Constrained Delegation on the Keyfactor Command Service Account 2828
Figure 532: Certificate Profile for EJBCA Client Certificate 2831
Figure 533: Certificate Download for EJBCA Client Certificate 2831
Figure 534: Microsoft CA Permissions 2833
Figure 535: EJBCA Access Permissions 2838
Figure 536: Add Client Certificate as Member of EJBCA Access Rule 2838
Figure 537: Keyfactor Command Service 2839
Figure 538: Include Expired and Revoked Certificates in Certificate Search 2840
Figure 539: Configure Expiration Renewal Handler: Add New Identity 2842
Figure 540: Configure Expiration Renewal Handler: Assign Role to Identity 2843
Figure 541: Keyfactor CA Policy Module Policy Module Handler Ordering 2848
Figure 542: Default Policy Module 2849
Figure 543: Install RFC 2818 Policy Handler: Begin Setup Wizard 2850
Figure 544: Install RFC 2818 Policy Handler: Select Components 2851
Figure 545: Enable the Keyfactor CA Policy Module 2852
Figure 546: Upload the Keyfactor CA Policy Module License 2853
Figure 547: Enable the RFC 2818 Policy Handler 2854
Figure 548: Add Templates for Management with the RFC 2818 Policy Handler 2855
Figure 549: Install SAN Attribute Policy Handler: Begin Setup Wizard 2856
Figure 550: Install SAN Attribute Policy Handler: Select Components 2857
Figure 551: Enable the Keyfactor CA Policy Module 2858
Figure 552: Upload the Keyfactor CA Policy Module License 2859
Figure 553: Enable the SAN Attribute Policy Handler 2860
Figure 554: Add Templates for Management with the SAN Attribute Policy Handler 2861
Figure 555: Install Whitelist Policy Handler: Begin Setup Wizard 2862
Figure 556: Install Whitelist Policy Handler: Select Components 2863
Figure 557: Enable the Keyfactor CA Policy Module 2864
Figure 558: Upload the Keyfactor CA Policy Module License 2865
Figure 559: Enable the Whitelist Policy Handler 2866
Figure 560: Add Templates for Management with the Whitelist Policy Handler 2867
Figure 561: Add Machines for Management with the Whitelist Policy Handler 2868
Figure 562: Keyfactor CA Policy Module NLog.config File 2870
Figure 563: Logi web.config 2871
Figure 564: Logi Configuration Settings—Keyfactor Command Portal Tab 2872
Figure 565: Logi Configuration Settings—Keyfactor Command Dashboards and Reports Tab 2873
Figure 566: Orchestrator Job Flow 2878

Figure 567: Select the Download x64 Option Under Run Console Apps 2881
Figure 568: Client Secret for Orchestrator Client in Keyfactor Identity Provider 2887
Figure 569: Local Security Policy 2888
Figure 570: CA Permissions 2890
Figure 571: Microsoft Certificate Template Application Policies for Client Authentication Certificate 2892
Figure 572: Microsoft Certificate Template Request Handling for Client Authentication Certificate 2893
Figure 573: Installation Files Blocked after Download 2898
Figure 574: CA Configuration Settings 2939
Figure 575: View Packages as Part of a List 2941
Figure 576: View Packages on Individual Pages 2941
Figure 577: Find the Latest Version of the Package 2942
Figure 578: Download the Package Zip File 2942
Figure 579: Universal Orchestrator on Windows NLog.config File 2952
Figure 580: Universal Orchestrator on Linux NLog.config File 2953
Figure 581: Universal Orchestrator Service 2954
Figure 582: Change Service Account Password in Services MMC 2955
Figure 583: Application Settings for Client Certificate Renewal 2965
Figure 584: Keyfactor Command Permissions Required for Automatic Renewal and Revocation of Client Authentication Certificates 2967
Figure 585: Search for System Environment Variables 2972
Figure 586: Edit the System Path Environment Variable to Add the Path to Java 2973
Figure 587: Add JAVA_HOME System Environment Variable 2974
Figure 588: Keyfactor Java Agent Local Installation on Windows 2978
Figure 589: Keyfactor Java Agent Local Installation on Linux 2983
Figure 590: Configure Logging for Keyfactor Java Agent on Windows 2988
Figure 591: Configure Logging for Keyfactor Java Agent on Linux 2989
Figure 592: Keyfactor Java Agent Service on Windows 2990
Figure 593: SSH Key Discovery Flow 2991
Figure 594: SSH User Key Management Flow 2992
Figure 595: Find the Server Group ID 2997
Figure 596: Configure Logging for the Keyfactor Bash Orchestrator 3003
Figure 597: Orchestrator Management for a Keyfactor Bash Orchestrator 3004
Figure 598: Orchestrator Management for a Keyfactor Bash Orchestrator 3004
Figure 599: Status for the Keyfactor Bash Orchestrator Service 3010
Figure 600: Certificate Incorrectly in the Trusted Root Certificate Store 3018
Figure 601: Find the Certificate for the Keyfactor Command Web Site 3020
Figure 602: Configure Keyfactor Command for Client Certificate Authentication 3029
Figure 603: IIS Module for Client Certificate Authentication 3030
Figure 604: Configure only Anonymous Authentication at the Server Level in IIS 3031
Figure 605: Disable Authentication Methods at the Application Level in IIS 3031
Figure 606: Configure SSL Settings in IIS for Client Certificate Authentication 3032
Figure 607: Configure IIS Client Certificate Mapping Authentication for the Default Web Site 3033
Figure 608: Configure Authorization Credentials for Keyfactor Orchestrators 3033
Figure 609: Configure Application Setting in Keyfactor Command to use the Header Certificate 3034
Figure 610: Client Certificate Authentication with AD Storage Does Not Require Certificate Authentication Configuration in Keyfactor Command 3036
Figure 611: IIS Module for Client Certificate Authentication with AD Storage 3037
Figure 612: Configure Client Certificate Authentication at the Server Level in IIS 3038
Figure 613: Disable Authentication Methods at the Application Level in IIS 3038
Figure 614: Configure SSL Settings in IIS for Client Certificate Authentication 3039
Figure 615: Microsoft Certificate Template General for Client Authentication Certificate 3040
Figure 616: Microsoft Certificate Template Request Handling for Client Authentication Certificate 3041
Figure 617: Microsoft Certificate Template Application Policies for Client Authentication Certificate 3042
Figure 618: Microsoft Certificate Template Security for Client Authentication Certificate 3043
Figure 619: System Environment Variable to Define a Proxy URL for Use by the Universal Orchestrator on Windows 3050
Figure 620: Configuration Wizard Route Information for the Keyfactor Portal 3057
Figure 621: Error During Upgrade 3075

Figure 622: System Alerts 3079
Figure 623: Example Navigation Menu Before Upgrade to 9.0 3133
Figure 624: Example Navigation Menu After Upgrade to 9.0 3133
Figure 625: New Risk Header 3135
Figure 626: Template Level Metadata 3136
Figure 627: Navigate Forward and Backwards Through Pages 3137
Figure 628: Keyfactor Logi License Expiration Alert 3158
Figure 629: Keyfactor Logi License Expiration Alert on the Dashboard 3158
Figure 630: Keyfactor Logi License Expiration Alert on Report 3159
Figure 631: Keyfactor Expired Logi Error Message 3159
Figure 632: Entry of gMSA Users in the Administrative Users Field 3162

1.0 Introduction

The *Keyfactor Command Documentation Suite* includes:

- *Keyfactor Command Reference Guide*
- *Keyfactor API Reference Guide*
- *Keyfactor Command Server Installation Guide*
- *Keyfactor Orchestrators Installation and Configuration Guide*
- *Keyfactor Command Release Notes & Upgrading*

In addition, Keyfactor offers documentation for products that are not part of the *Keyfactor Command Documentation Suite*, including the *Keyfactor Command Upgrade Overview* and installation guides for third-party CA gateways that interface with Keyfactor, which are available upon request.

2.0 Reference Guide

The *Reference Guide* for the Keyfactor Command solution by Keyfactor provides comprehensive instructions on using the Keyfactor Command Management Portal and Policy Module. The Management Portal is the command and control center for Keyfactor Command. From here, you can get a quick glance at the health of your PKI and a sense of how it is being used by visiting the dashboard, or delve into details of certificates using the certificate search feature. The Management Portal is also used to configure workflow and email notifications, enroll for certificates, and configure options that are used across the whole of the Keyfactor Command product.

This reference guide covers advanced configuration of Keyfactor Command in addition to providing usage information.

This guide is organized in the order of the Management Portal menu panel.

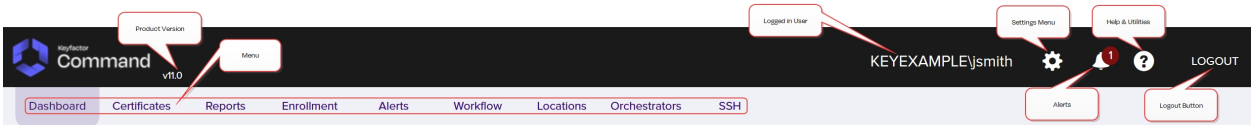


Figure 1: Management Portal Menu

2.1 Using the Management Portal

The Keyfactor Command Management Portal is a web-based application that you can open in any supported browser. The default URL for the Management Portal is (where KEYFACTOR_SERVER_FQDN is the FQDN of your Keyfactor Command administration server):

https://KEYFACTOR_SERVER_FQDN/keyfactorportal

In addition to the main URL, the pages in the Management Portal are available via deep link. To find the deep link for a page, just visit the page in your browser and copy the URL from the browser's URL line. For example, the deep link URL directly to the certificate search page in the Management Portal is available at:

https://KEYFACTOR_SERVER_FQDN/keyfactorportal/CertificateCollection/Edit?cid=0

You can change the number at the end of this deep link to direct the deep link to a specific saved collection instead of the main search. You can find the collection number by browsing to the collection and viewing the URL in your browser. You can also build links to specific searches, rather than saved collections. For more information, see Certificate Search and Collections on page 19.

The following is some information to help you understand and use the Management Portal successfully.

Navigating Keyfactor Command Grids

The grid includes the following features:

- **Action buttons** are used to perform actions on the data in the rows displayed in the grid. Some buttons are grayed out until you click on a grid row, or if that action is unavailable for the selected row. Which action buttons are displayed will depend on the function of the page.

  Note: On some grids the actions are also available from the context menu, which is accessible by right-clicking on the selected row.

- The **Total** in the upper right of the grid will be updated each time you refresh the grid.
- The **Refresh** button will poll the Keyfactor Command database and update the grid with the results of the current page query and update the Total.
- To change a **column width**, click, hold and drag the line separating two column headers (to the right of the column you want to change).
- To **rearrange columns**, click on the header of the column you want to move and hold and drag the column to your selected location.
- To change the **sort order** of the grid, click on the header of the column you wish to sort by. The first time you click, the grid will be sorted in ascending order by the selected column. Click the column header again to reverse the sort order. When a column is sorted, a purple caret will appear at the end of the column name showing the direction of the sort. Lack of a caret indicates the grid is sorted by the default column and order. On some grids only select columns are sortable.
- Click anywhere on the **row**, or on the tick box in the far left column of a grid row, to select that row. You may select multiple rows by utilizing the standard Windows selection functions of CRTL/Select and SHIFT/Select to select multiple rows at once. Selected rows will be highlighted purple. You may then perform actions on the selected row(s) depending on the functionality of the grid by right-clicking and selecting an action (if available) or selecting an action from the action buttons at the top of the grid. Tick boxes are found only on grids that support actions on multiple rows at once.
- Information in a grid field can be **copied** to the clipboard by highlighting text in a grid field and clicking **Ctrl**+**C**.
- Hovering over a row will change the row green to show which row the cursor is focused on.
- To open up the details pop-up for a row, or a search page, depending on the functionality of the screen, double click on a row, or select the row and then select an action button from the grid header or the context menu item, if available, by right-clicking.
- Grids use scroll bars to display grids with large quantities of data.
- Grid pages will re-size with the window size.

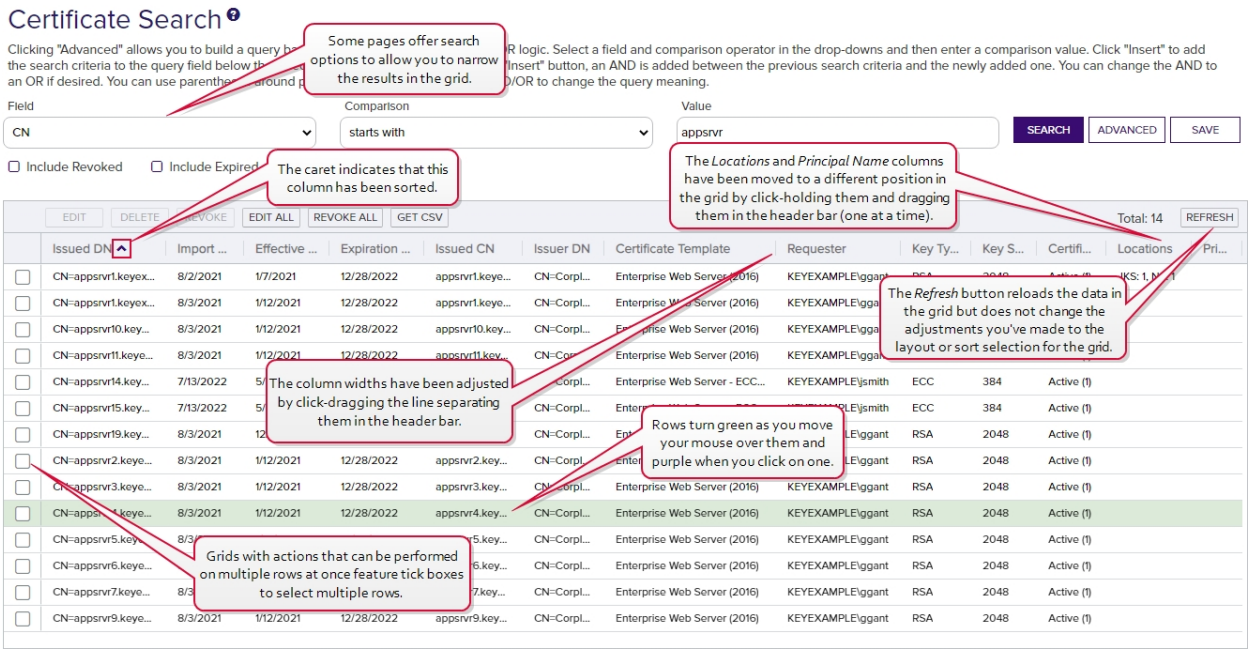


Figure 2: Using the Management Portal Grids

Validating Data Types

In data entry dialogs, fields in which the user is expected to enter certain data types will validate the user input against the expected data type and produce an error if the data entered is not valid. Fields that typically have validation include:

- email addresses (string fields)
- integers
- strings (alpha-numeric)

Further to this, regular expressions are supported on select entry fields for enrollment (see Certificate Template Operations on page 381).

Pop-up Dialogs

- When the cursor is focused on a field, the outline of the field will turn purple.
- Active/ available buttons will be bright purple. Inactive/ unavailable buttons will be faded to light purple. When data entered into the panes changes the conditions, the buttons may change between bright and light purple (active/inactive).
- At the bottom of most pop-up dialogs are the Save and Cancel buttons, and possibly other actions that can be performed on the data, depending on the purpose of the pane.
- The *X* in the top right corner is the close option which works like the cancel button.

- Many pop-up panes will have multiple tabs. The tab in which the cursor is focused will be underlined in green. When you point the cursor at another tab, it will temporarily change the underlining to green until you click into the tab.

Under Construction Icon

The under construction icon will display when an action of a transaction is *in process*.

Figure 3: Under Construction Icon

Confirmation Message

Messages appear at the bottom of the screen during processing at times. For example, an operation successful message will appear at the bottom of the screen when a selected action on a transaction is successful.

Figure 4: Confirmation Message

2.1.1 Authentication and Authorization

Keyfactor Command can be configured to use either Active Directory as an identity provider or an identity provider other than Active Directory. If you choose to use Active Directory as an identity provider, Keyfactor Command is by default configured to support both Windows integrated authentication and Basic authentication. Windows integrated authentication allows users on domain-joined computers using domain accounts and browsers configured to support integrated authentication to access the Keyfactor Command Management Portal without needing to provide a username or password to authenticate to the Management Portal or Keyfactor API endpoints (from the Keyfactor API Reference and Utility accessed in the same browser session) assuming they have a valid Kerberos ticket. Keyfactor Command can be configured to support only Basic authentication, which requires entry of a username and password to authenticate to the Management Portal or Keyfactor API endpoints. This can be useful in environments where integrated authentication is not practical or desired, such as when users access the Management Portal using different accounts than they use to log on to their computers.

When using an identity provider other than Active Directory, Keyfactor Command uses tokens for authentication and requires users to enter a username and password to authenticate to the Management Portal. To use the Keyfactor API, users need to acquire a token (see Authenticating to the Keyfactor API on page 844).

Keyfactor Command uses a system of security roles and claims to provide access control to the Management Portal as a whole and to the features within it and the Keyfactor API. In order to access

the Management Portal or Keyfactor API, the account you are using to access Keyfactor Command must be a member of one of the groups granted access to the Management Portal during the Keyfactor Command installation and configuration process (see Administrative Users Tab on page 2805) or your account must have been granted access either directly or via group membership later through the Management Portal (see Security Roles and Claims on page 622) or with the Keyfactor API (see Security Roles on page 2081).

2.1.2 Dashboard

The dashboard, at the top level of the Management Portal, provides you with a quick glance at the status of your PKI. It is a global representation of your PKI and does not filter data based on your access.

Risk Header

The top of the page shows a risk header, which is made up of six sticky notes displaying active certificates, expiring and expired certificates, revoked certificates, and certificates with weak keys. The dashboard risk header displays by default and cannot be moved or removed (though it may be hidden with a security setting).

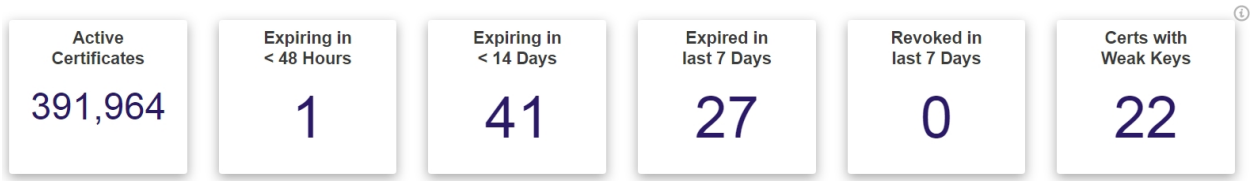


Figure 5: Dashboard Risk Header

The risk header panels are:

- Active Certificates

  This value reflects all active certificates in the database, including those with a certificate state of *unknown*, and excludes expired and revoked certificates.

- Certificates Expiring in Less Than 48 Hours

  This value includes all active certificates in the database with an expiration date between the current date/time and 48 hours from the current date/time.

- Certificates Expiring in Less Than 14 Days

  This value includes all active certificates in the database with an expiration date between the current date/time and 14 days (to the minute) from the current date/time. This value includes certificates shown in the *Expiring in < 48 Hours* panel.

- Certificates Expired in the Last 7 Days

  This value includes all certificates that have expired within the previous 7 days. This is the only panel that includes expired certificates.

- Certificates Revoked in the Last 7 Days

This value includes all certificates that have been revoked within the previous 7 days. This is the only panel that includes revoked certificates.

- Certificates with Weak Keys

  This value includes all certificates in the database that are deemed to have weak keys. Weak key certificates are those with signature algorithms SHA-1, MD5, RSA key size less than 2048, and ECC key size less than 224.

Tip: Access control to the risk header is controlled separately from the dashboard page as a whole, so a user could be granted access to the dashboard but not to the risk header and in this way see a dashboard that did not display the risk header. For more information, see Security Role Permissions on page 632.

Customizable Panels

A variety of panels are available to add to the dashboard, including:

- A separate panel for each of your certificate authorities (CAs) configured for synchronization can be displayed with graphs showing the activity over the last X weeks (24 by default) and a pie chart showing all active certificates by template. The number of weeks to display is configurable on a panel-by-panel basis. See Dashboard: CA Status on page 11.

  Note: Any CAs that have not been configured for synchronization will not appear as available for addition on the dashboard, or for reports which require selecting a CA.

- Certificate collections (see Certificate Collection Manager on page 85) can be configured to be included in a bar chart on the Certificate Collection dashboard panel. See Dashboard: Collections on page 12.
- The Certificates by Signing Algorithm panel displays a bar chart showing all active certificates broken down by signing algorithm. The CAs to include in the display are configurable. Both CAs that are currently configured for synchronization and any that were previously synchronized are available for inclusion. Certificates imported into Keyfactor Command via SSL scanning, certificate store inventorying, and manual import are also included and can be filtered out by unchecking the *Certificates Not Associated with CA* option. See Dashboard: Certificates by Signing Algorithm on page 13.
- The Recent Certificate Store Jobs panel displays the status of up to ten jobs. Both completed and in progress jobs are included. See Dashboard: Recent Certificate Store Jobs on page 15.
- If you configure certificate revocation list (CRL) or online certificate status protocol (OCSP) locations for monitoring and opt to display them on the dashboard (see Revocation Monitoring on page 210), these will appear with a status on the dashboard Revocation Monitoring panel. See Dashboard: Revocation Monitoring on page 16.
- The comprehensive SSL Endpoints panel includes a grid of changes found in existing SSL endpoints, a grid of endpoints with certificates expiring in the next X days, a pie chart showing SSL endpoints per defined SSL network, and a pie chart showing the results from the

last SSL scan broken out by result (e.g. certificate found, connection timed out, connection refused). The number of days for the expiring certificates grid is configurable. See Dashboard: SSL Endpoints on page 17.

- The status of SSL discovery and monitoring jobs can be displayed on an orchestrator-by-orchestrator basis on the SSL Orchestrator Job Status panel. The orchestrators to include are configurable. See Dashboard: SSL Orchestrator Job Status on page 18.
- The Number of SSH Keys per Type panel includes SSH keys found on discovery and those issued through the Management Portal and displays as a bar chart broken down by key type. See Dashboard: Number of SSH Keys per Type on page 14.

The panels on the dashboard are displayed in two columns. You can click and drag the dividing line between the two columns to change the width of the columns—for example, a wide left column and a narrower right column. The panels can be rearranged by dragging them up and down a column or from one column to the other. If you've chosen to change the column widths, you can arrange the wider panels in your wider column and the narrower panels in your narrower column.

The selected panels and their arrangement is unique to each user of the Management Portal. Out of the box, in addition to the risk header, the dashboard includes the Collections and Revocation Monitoring panels, so each new user to the dashboard will see these panels.

The latest version of the Logi reporting engine has functionality which avoids a system timeout issue by periodically pinging the IIS session behind the scenes so that the dashboard doesn't time out when the session has been idle. As a result, the dashboard no longer refreshes after 20 minutes, but invokes this new functionality instead. The settings used to control this depend on the **Session State Timeout** and **Session Auto Keep Alive** attribute settings in IIS. For more information on this see:

https://devnet.logianalytics.com/hc/en-us/articles/1500009515942-Manage-Session-Timeout

Add a Panel to the Dashboard Display

To add a panel for display on your dashboard:

1. Click the Add Panel button on the left just below the dashboard risk header.

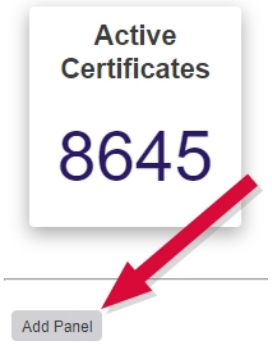


Figure 6: Click the Dashboard Add Panel Button

2. On the Add Panels dialog, select the panels you wish to display on the dashboard, click **Add** and then click **Done** at the bottom of the dialog.

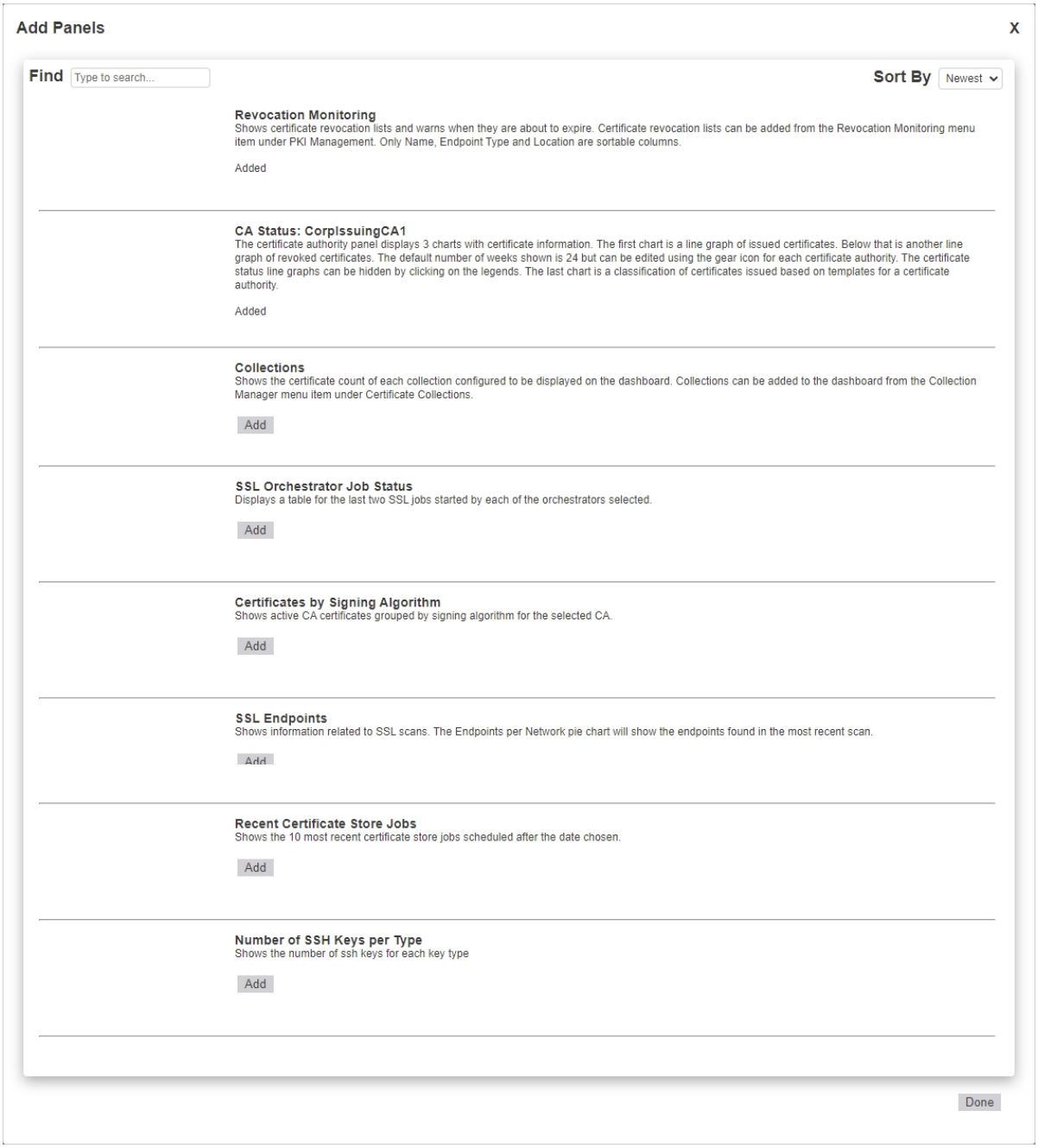


Figure 7: Add Panels to the Dashboard

Rename a Dashboard Panel

The panels displayed on the dashboard may be given user-defined names. To rename a displayed panel:

1. Click the panel **Settings** icon on the right of the panel you wish to rename and then click **Rename**.

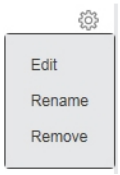


Figure 8: Dashboard Panel Settings

2. In the title field of the panel, type a new name. Click away from the field to save.

Figure 9: Type in a New Name for the Panel

Note: Only letters, numbers, spaces, and select punctuation marks are supported in the panel name field. Special characters, such as < and > (and therefore HTML markup), are not supported.

Remove a Dashboard Panel

To remove a panel from display on your dashboard:

1. Click the panel **Settings** icon on the right of the panel you wish to remove and then click **Remove**.

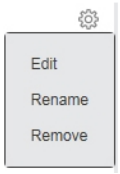


Figure 10: Dashboard Panel Settings

2. When prompted, confirm that you are sure that you want to remove the panel.

Tip: The **Edit** option only appears on the panel settings menu for selected panels.

2.1.2.1 Dashboard: CA Status

Each CA section of the dashboard includes two line graphs showing issued (top graph) and revoked and failed/denied certificate requests (bottom graph) over the last X weeks or days (24 weeks by default) on the left and a pie chart showing all active certificates by template on the right. To change the number of weeks displayed on the line graphs for **all** CAs, change the *Weeks of CA Stats* application setting (see Application Settings: Console Tab on page 602). To change the number of weeks or days displayed on the line graph on a CA-by-CA basis, click the panel **Settings** icon for the selected CA and choose **Edit**. A maximum of 52 weeks or 30 days may be configured when setting the time frame on a CA-by-CA basis.

The panel is interactive in a number of ways:

- Hover over a point on a line graph to see details for that point.
- Click on a point on a line graph to be taken to a new window with the certificate search page populated by the query of the selected CA and date.
- Click on a legend (e.g. Revoked) below a line graph to toggle add/remove that line from the chart.
- Click one of the labels below the pie chart to toggle add/remove that segment of the pie from the chart. This can be helpful, for example, if you remove a template that makes up the bulk of the chart, allowing you to just focus on the remaining templates (and making these pie segments bigger and easier to click on).
- Hover over a number for, or section of, the pie chart to see the template name associated with that section of the pie chart. This is the number of active certificates for that template.
- Click on a number for, or section of, the pie chart to be taken to a new window with the certificate search page populated by the query of the selected CA and template.

A status indicator appears at the top of the CA section showing when the CA was last contacted.

Click the **Hide** button to minimize the display. Click the panel **Settings** icon to remove or rename the panel or change the comparison date for the display (see Dashboard on page 6). Data for the CA sections of the dashboard is generated from certificates retrieved during CA synchronization tasks (see Certificate Authorities on page 349).

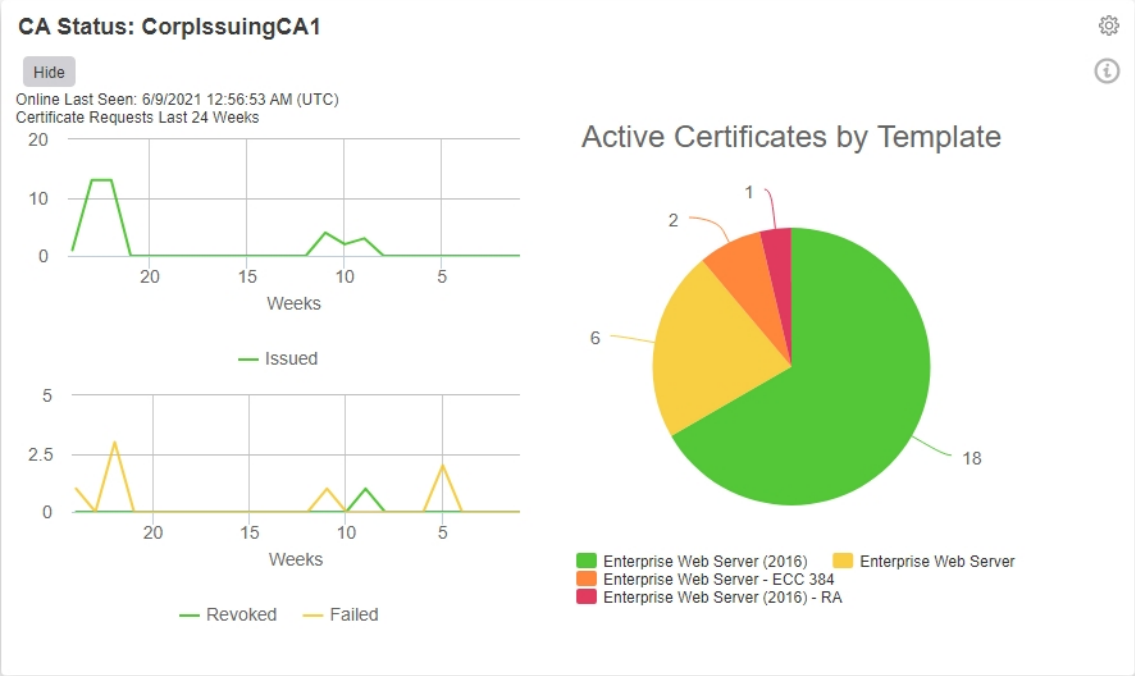


Figure 11: Dashboard CA Snapshot

2.1.2.2 Dashboard: Collections

If you opt to include any certificate collections for display on the dashboard (see Certificate Collection Manager on page 85), you will see the data on the Collections dashboard panel. This panel shows a bar representing the total number of active, expired and revoked certificates for each certificate collection configured for dashboard display. Hover over a bar to see the number of certificates in the collection. Click on a bar to open the certificate search page in a new window filtered for that certificate collection.

Note: The collections dashboard widget will only display the first 25 collections alphabetically.

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

Click the **Hide** button to minimize the display. Click the panel **Settings** icon to remove or rename the panel or change the comparison date for the display (see Dashboard on page 6).

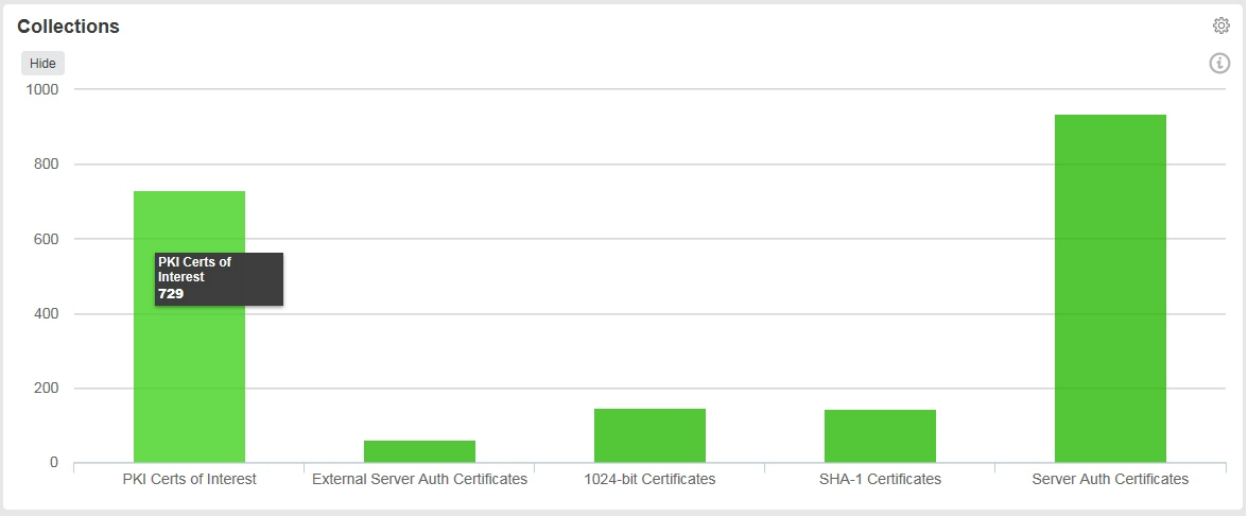


Figure 12: Dashboard Certificate Collections

2.1.2.3 Dashboard: Certificates by Signing Algorithm

The Certificates by Signing Algorithm panel on the dashboard shows a bar chart of all active certificates synchronized to Keyfactor Command from a Microsoft CA or Keyfactor CA gateway or imported via SSL scanning, certificate store inventorying, or manual import broken down by signing algorithm. Hover over a bar to see the number of active certificates in the category. By default, all certificates in the Keyfactor Command database are included. To include only selected CAs or gateways, click the panel **Settings** icon and choose **Edit**. In the Edit dialog, select the CAs you wish to include in the panel. To filter out certificates brought into the database via SSL scanning, certificate store inventorying, and manual import, select specific CAs and uncheck the *Certificates Not Associated with CA* option.

Click the **Hide** button to minimize the display. Click the panel **Settings** icon to remove or rename the panel or change the comparison date for the display (see Dashboard on page 6).

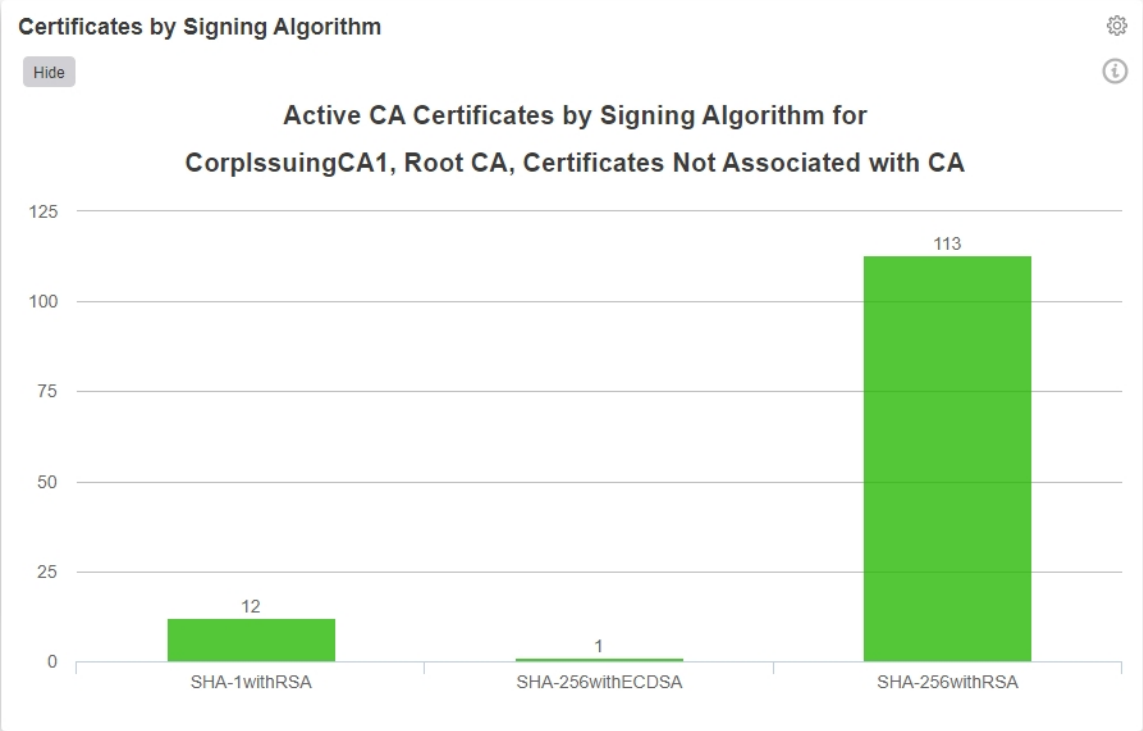


Figure 13: Dashboard Certificates by Signing Algorithm

2.1.2.4 Dashboard: Number of SSH Keys per Type

The Number of SSH Keys per Type panel on the dashboard shows a bar chart of all SSH keys in the Keyfactor Command database. The chart includes both managed keys (those generated within Keyfactor Command using My SSH Key (see My SSH Key on page 531) or the service account key page (see Service Account Keys on page 542) and unmanaged keys (see Unmanaged SSH Keys on page 556). Hover over a bar to see the number of SSH keys in the category.

Click the **Hide** button to minimize the display. Click the panel **Settings** icon to remove or rename the panel or change the comparison date for the display (see Dashboard on page 6).

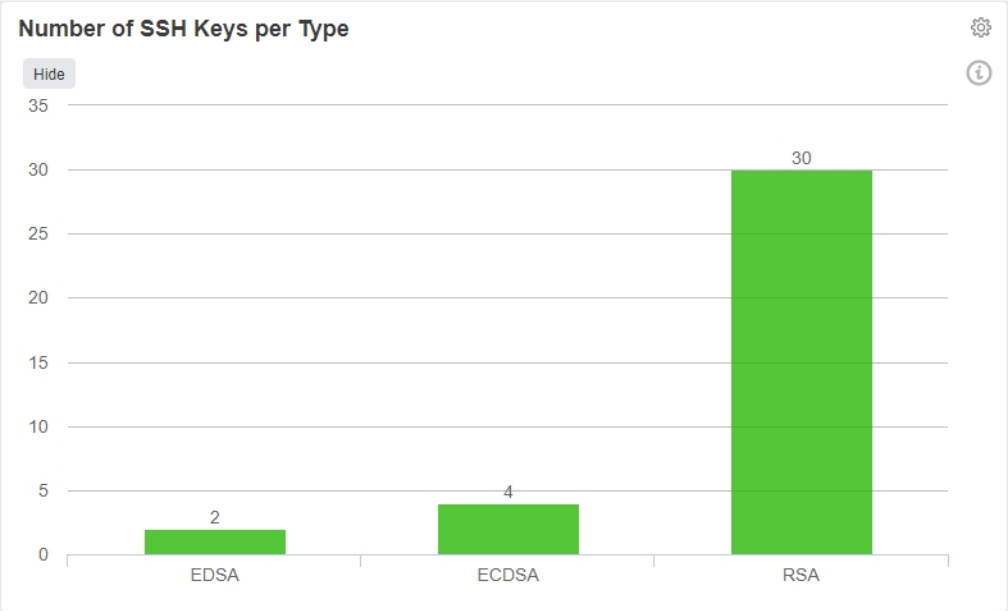


Figure 14: Dashboard SSH Keys per Type

2.1.2.5 Dashboard: Recent Certificate Store Jobs

The Recent Certificate Store Jobs panel on the dashboard includes a grid showing the most recent job history for certificate stores. Both completed (successful or not) and in progress jobs are included. The grid includes the orchestrator name, the target for the job (which in most cases includes the host name and the certificate store name), the job start date, the job type (e.g. inventory or management for an IIS or F5 store), and color-coded results (errors appear in red) for the job.

Click on the name of the orchestrator in the grid to be taken to the orchestrator job history page with the query populated by the selected orchestrator.

To include only jobs that started on or after a selected date, click the panel **Settings** icon and choose **Edit**. In the Edit dialog, either enter a comparison date or use the calendar picker to select a date. Only jobs with a starting date on or after this date will be shown. A maximum of ten jobs are shown.

Click the **Hide** button to minimize the display. Click the panel **Settings** icon to remove or rename the panel or change the comparison date for the display (see Dashboard on page 6).

Recent Certificate Store Jobs

Hide

Orchestrator Name	Target	Start Date (UTC)	Job Type	Result
k8s-universal-orchestrator-splus-1	ns3.keyexample.com - /nsconfig/ssl	9/18/2023 3:59:00 PM	CitrixAdcInventory	Success
k8s-universal-orchestrator-splus-1	ns3.keyexample.com - /nsconfig/ssl	9/18/2023 3:58:00 PM	CitrixAdcInventory	Success
websrvr83-4Aug.keyexample.com	bigip16.keyexample.com - Common	9/18/2023 3:57:00 PM	F5-SL-RESTInventory	Success
k8s-universal-orchestrator-splus-1	ns3.keyexample.com - /nsconfig/ssl	9/18/2023 3:57:00 PM	CitrixAdcInventory	Success
websrvr83-4Aug.keyexample.com	bigip16.keyexample.com - Common	9/18/2023 3:54:00 PM	F5-SL-RESTInventory	Success
websrvr83-4Aug.keyexample.com	bigip16.keyexample.com - Common	9/18/2023 3:51:00 PM	F5-SL-RESTInventory	Success
websrvr21-9Aug.keyexample.com	appsrvr76.keyexample.com - /opt/app/ServerCertificate1.pem	9/18/2023 3:45:00 PM	RFPEMInventory	Failure
websrvr21-9Aug.keyexample.com	appsrvr76.keyexample.com - /opt/app/ServerCertificate1.pem	9/18/2023 3:30:00 PM	RFPEMInventory	Failure
websrvr21-9Aug.keyexample.com	appsrvr76.keyexample.com - /opt/app/ServerCertificate1.pem	9/18/2023 3:15:00 PM	RFPEMInventory	Failure
k8s-universal-orchestrator-splus-1	bigip16.keyexample.com - Common	9/16/2023 12:19:00 AM	F5-SL-RESTInventory	Success

Figure 15: Dashboard Recent Certificate Store Jobs

2.1.2.6 Dashboard: Revocation Monitoring

The Revocation Monitoring panel on the dashboard shows each configured CRL and OCSP location (if they have been configured to appear on the dashboard) with the path to the CRL or OCSP, the publication, next publish date, and expiration dates of the CRLs (these aren't relevant for OCSPs) and the status of the CRL or OCSP. The status for a CRL will show *Warning* if the expiration date of the CRL is within the warning period as defined by the number of weeks, days, or hours configured in the *Show on Dashboard* setting (see Revocation Monitoring Location Operations on page 211). For example, if you had a CRL that expired on June 30 and configured the warning period to 15 days before expiration, the Warning status would begin to appear on the dashboard for that CRL on June 15.

Some columns allow for sorting in ascending or descending order by clicking the column heading to toggle sort order. Click the **Hide** button to minimize the display. Click the panel **Settings** icon to remove or rename the panel or change the comparison date for the display (see Dashboard on page 6).

Revocation Monitoring

Hide

Name	Endpoint Type	Location	Publication (UTC)	Next Publish by Date	Expiration (UTC)	Status
Issuing One	CRL	http://www.keyexample.com/CorpIssuing1.crl	6/17/2020, 4:56:02 PM	7/29/2020, 5:06:02 PM	8/8/2020, 5:16:02 PM	Expired
Issuing Two	CRL	http://www.keyexample.com/CorpIssuing2.crl	6/16/2022, 7:49:57 PM	7/14/2022, 7:59:57 PM	7/28/2022, 8:09:57 PM	Valid
Issuing Three	CRL	http://www.keyexample.com/CorpIssuing3.crl	6/24/2022, 6:09:31 PM	7/24/2022, 6:19:31 PM	7/31/2022, 6:29:31 PM	Warning
Root One	CRL	http://www.keyexample.com/CorpRoot1.crl	5/14/2022, 4:41:05 PM	11/14/2022, 4:51:05 PM	12/15/2022, 5:01:05 PM	Valid
Root Two	CRL	http://www.keyexample.com/CorpRoot2.crl	8/7/2021, 12:30:22 AM	4/7/2022, 12:40:22 AM	5/8/2022, 12:50:22 AM	Expired
Issuing One	OCSP	http://websrvr75.keyexample.com/ocsp				Valid
Issuing Two	OCSP	http://websrvr75.keyexample.com/ocsp				Valid
Issuing Three	OCSP	http://websrvr75.keyexample.com/ocsp				Valid

Figure 16: Dashboard Revocation Monitoring Status

2.1.2.7 Dashboard: SSL Endpoints

The comprehensive SSL Endpoints panel includes several components:

- The Changes Found to Existing Endpoints grid displays up to ten SSL endpoints for which a change was found in the most recent scan from the previous scan status. The grid includes the endpoint address, scan time, and both the previous and current endpoint status. This grid only displays if there are endpoints that have been changed.
- The Endpoints Expiring in the Next X Days grid displays up to ten SSL endpoints with certificates expiring in the next X days. This grid only displays if there are endpoints that meet that criteria. If there are more than ten to display, the certificates expiring soonest are displayed. Out of the box, the number of days is configured to 30. To change the number of days, click the panel **Settings** icon, choose **Edit**, enter a number of days, and click **Done**. To clear the custom number of days and return to the default, click the panel **Settings** icon, choose **Edit**, clear the days field, and click **Done**. The grid includes the network name, the endpoint address, the certificate expiration date, and the certificate common name, if any.
- The Endpoints per Network pie chart shows discovered SSL endpoints broken down by SSL network. All discovered endpoints are included. This includes endpoints at which a certificate is currently being found, endpoints at which a certificate was found in the past but is no longer found, and endpoints that responded in some way on scan but did not present a certificate. Click on a section of the pie chart to be taken to the SSL Discovery Results page. Click any of the labels below the pie chart to toggle add/remove that segment of the pie from the chart.
- The Network Endpoint SSL Scanning Results pie chart shows the results from the most recent SSL scan (discovery or monitoring) broken out by result (e.g. certificate found, connection timed out, connection refused). Click on a section of the pie chart to be taken to the SSL Discovery Results page. Click any of the labels below the pie chart to toggle add/remove that segment of the pie from the chart. This can be helpful, for example, if you remove the certificate found section, allowing you to just focus on any errors (and making the error pie segments bigger and easier to click on).

Click the **Hide** button to minimize the display. Click the panel **Settings** icon  to remove or rename the panel or change the comparison date for the display (see Dashboard on page 6).

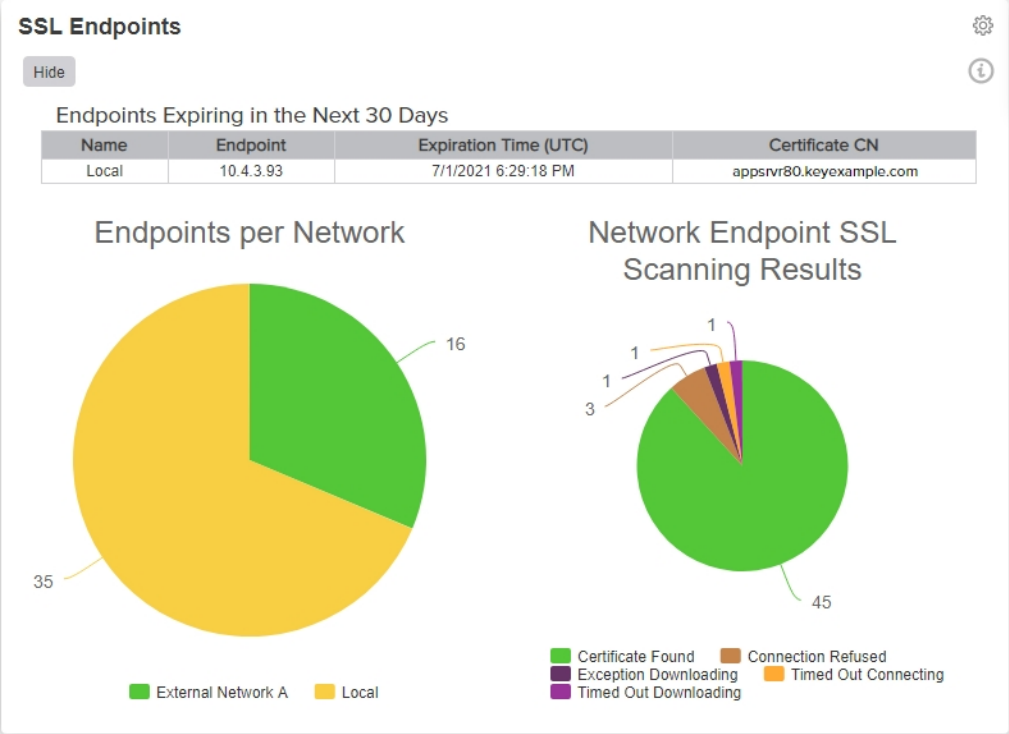


Figure 17: Dashboard SSL Endpoints

2.1.2.8 Dashboard: SSL Orchestrator Job Status

The SSL Orchestrator Job Status panel on the dashboard displays a grid showing the results of the two most recent SSL jobs for each active Keyfactor Universal Orchestrator with the SSL capability in the configured orchestrator pool (see Orchestrator Pools Definition on page 470). Both jobs in progress and completed jobs are included. The grid includes the names of the orchestrators in the selected pool(s), the job type, job start date and time, color-coded results (errors appear in red), and color-code status (jobs in progress are yellow). To change the orchestrator pools included in the display, click the panel **Settings** icon, choose **Edit**, select the desired orchestrator pools, and click **Done**.

Click on the name of an orchestrator in the grid to be taken to the orchestrator job history page with the query populated by the selected orchestrator.

Click the **Hide** button to minimize the display. Click the panel **Settings** icon to remove or rename the panel or change the comparison date for the display (see Dashboard on page 6).

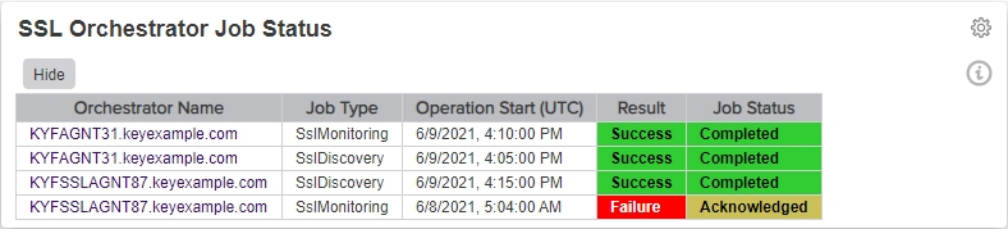


Figure 18: Dashboard SSL Orchestrator Job Status

2.1.3 Certificate Search and Collections

The Keyfactor Command database can include certificates from many locations—including certificates synchronized from your Microsoft CAs in both the primary forest and alternate forests, certificates synchronized from your EJBCA CAs, certificates synchronized from cloud-based certificate vendors via the Keyfactor certificate gateways, certificates automatically imported based on configured SSL endpoint locations (see SSL Discovery on page 453), certificates imported from certificate stores (see Certificate Stores on page 408), and manually imported certificates (see Add Certificate on page 74). The Certificate Search function allows you to query the database for certificates from any available source. You do not need to specify the source as part of the query.

A specific certificate search may be saved as a collection, which can then be revisited without needing to enter the search selections again. The saved collection can then be referenced from other parts of the Management Portal (e.g. expiration alerts, the dashboard, and select reports). Certificate collections may be added to the *Certificates* menu of the Management Portal for quick access. Several default certificate collections are created in new installations. For more information, see Certificate Collection Manager on page 85.

Note: The options shown and described in this section are available to full administrative users of the Management Portal. Users with limited access to the Management Portal will not see all the options (e.g. the recover buttons may not appear) and will see some slightly different buttons (e.g. the edit buttons shown may say *view* instead of *edit*).

2.1.3.1 Certificate Details

The cornerstones of the Keyfactor Command Management Portal are the Certificate Search and the Certificate Details pages. The Certificate Details page includes a comprehensive set of details about each certificate managed by Keyfactor Command. To access a certificate's details, double-click on a row of the certificate search grid, or highlight a row, right click and select **Edit** (**Display** for users with only Read permissions) from the action menu (see Certificate Search Page on page 34).

The following action buttons are conveniently located at the top of the Certificate Details page for users with the appropriate permissions: **Revoke**, **Download**, **Renew**. See Certificate Operations on page 45 for more information on these actions.

Content Tab

The Content tab shows the certificate attributes from the CA (Active Directory in the case of a Microsoft CA). These fields are not editable. The list of Subject Alternative Names (SANs) and SAN count are also included on this tab. For an ECC certificate, the elliptic curve algorithm is included on this tab.

Tip: Double-click any field on this dialog to open a pop-up showing just that detail.

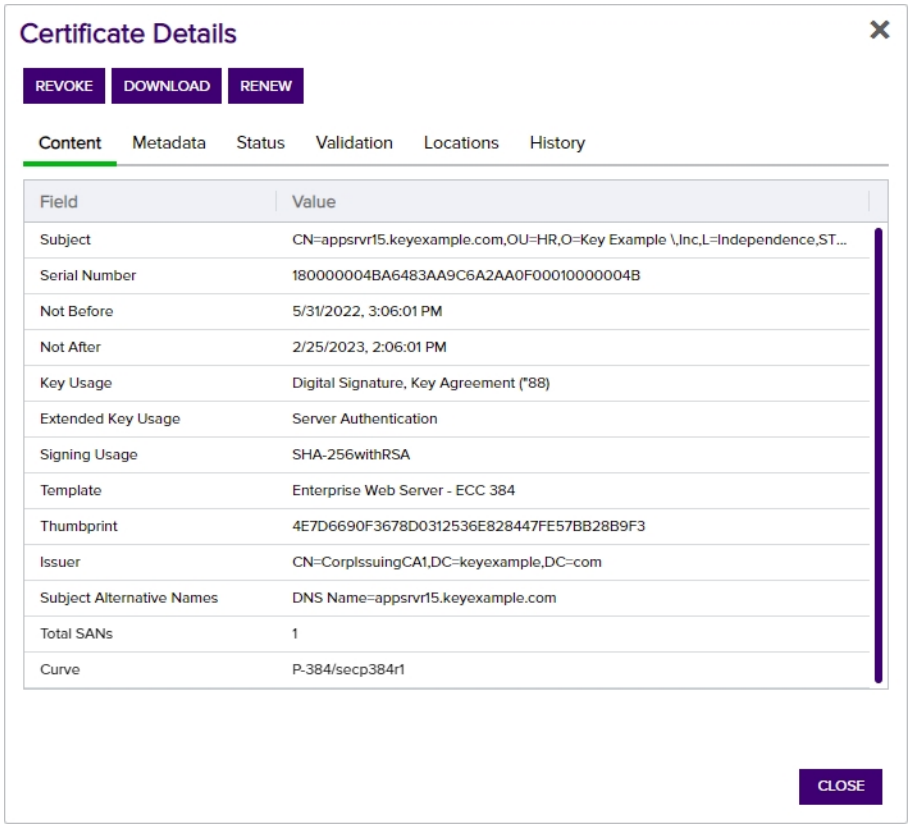


Figure 19: Certificate Details: Content Tab

Metadata Tab

The Metadata tab displays all metadata fields created for your Keyfactor Command implementation and shows any data in fields that have been populated with values specific to the certificate. Depending on the metadata type, these fields appear as text boxes, radio buttons, drop-downs, date fields, table or large text fields.

For users with edit permissions, on date fields a small popup calendar will appear that will allow you to select a date and will properly format it for you. You may edit values for any metadata fields to update the data at any time. You may also update multiple certificates' metadata with the same data by selecting multiple certificates from the certificates grid. Required fields will be marked with

*Required next to the field label. See Certificate Metadata on page 710 for information on this functionality.

Tip: The order of the metadata fields as they appear on this dialog is configurable using the certificate metadata display order option (see Sorting Metadata Fields on page 715).

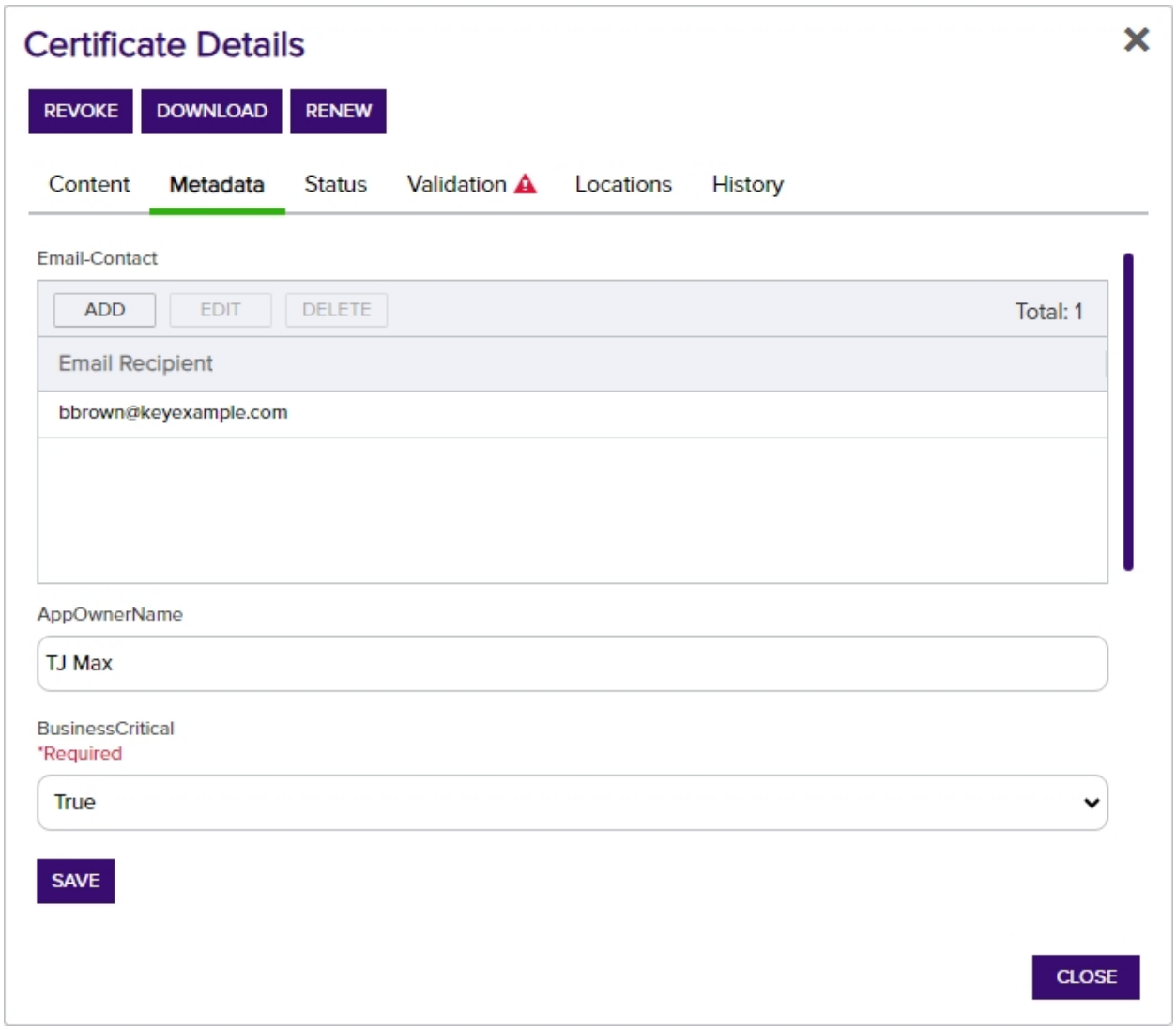


Figure 20: Certificate Details: Metadata Tab

The email metadata type will display a grid with the list of email addresses associated with the certificate(s). Grid action options of **Add**, **Edit** and **Delete** are included.

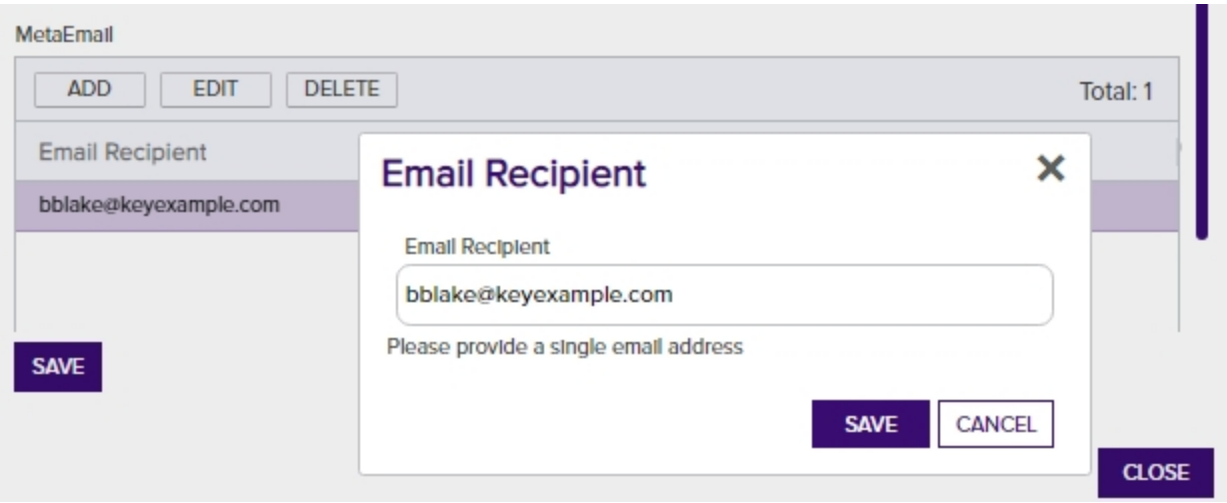


Figure 21: Email Type Metadata

Status Tab

The status tab displays some additional information about the certificate (see Table 1: Status Tab Descriptions).

The fields on this tab cannot be edited.

Tip: Double-click any field on this dialog to open a pop-up showing just that detail.

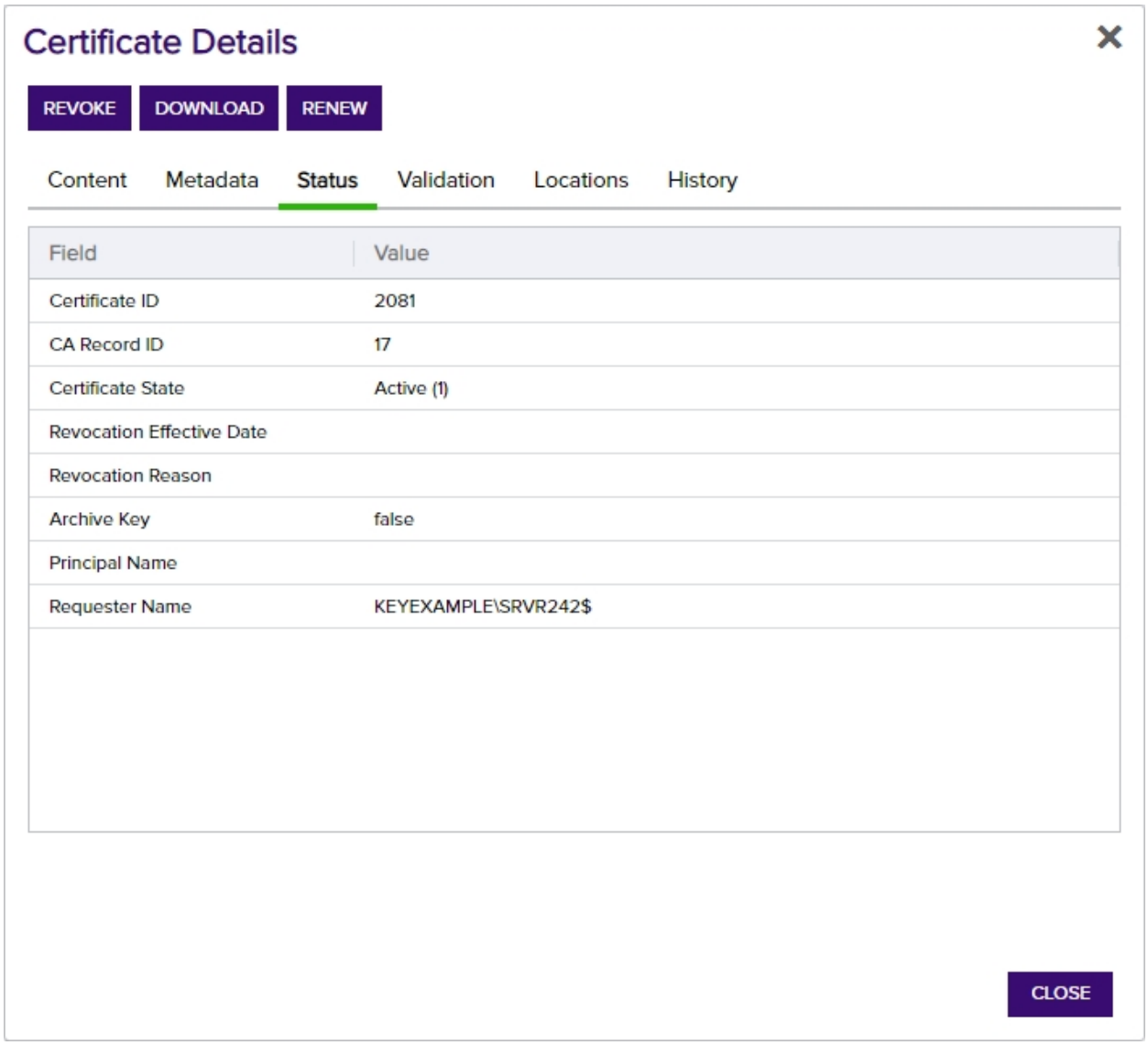


Figure 22: Certificate Details: Status Tab

Table 1: Status Tab Descriptions

Field	Description
Certificate ID	The Keyfactor Command reference ID for the certificate, which can be useful when referring to the certificate using API methods.
CA Record ID	The ID of the certificate in the CA (this has replaced CARequestID).
Certificate State	The state of the certificate. • Unknown (0)—This certificate entered the system in a manner other than a CA sync, so no status from a CA has been reported. • Active (1)—The "normal" state for certificates brought in via CA sync. The certificate has not been revoked. **Note:** Here we mimic the behavior of the Microsoft CA, which does not have a status for Expired, so certificates continue to be listed as Active or Revoked (as appropriate) after they expire. • Revoked (2)—The certificate has been revoked. • Failed (4)—The certificate has been denied approval. • Pending (5)—The certificate is awaiting approval. • Certificate Authority (6)—The certificate synced in from a CA sync that is indicated to be that CA's own certificate. • Parent Certificate Authority (7)—The certificate synced in from a CA sync that is indicated to be the certificate of a CA further up the chain. • Waiting for External Validation (8)—The certificate is pending, awaiting approval outside of Keyfactor Command. Generally, the certificate would have been added through one of the Keyfactor Command CA gateways using an EV certificate type.
Revocation Effective Date	If the certificate is revoked, the date it was revoked will be displayed here.
Revocation Reason	If the certificate is revoked, the reason will be displayed here. This is shown as a numeric value, which will be one of: • 0 — Unspecified • 1 — Key Compromised • 2 — CA Compromised • 3 — Affiliation Changed • 4 — Superseded • 5 — Cessation of Operation • 6 — Certificate Hold • 999 — Unknown

Field	Description
	See Revoke on page 71 for more information about revoking certificates.
Archive Key	If true, the certificate has a private key archived on the Microsoft CA to support CA key recovery. This flag is not an indicator for whether the certificate has a private key stored in Keyfactor Command. **Tip:** CA-level key recovery is supported for Microsoft CAs to allow recovery of private keys for certificates enrolled outside of Keyfactor Command. CA-level key archiving is not supported for enrollments done through Keyfactor Command. CA-level key recovery is not supported for EJBCA CAs. For enrollments done through Keyfactor Command for either Microsoft or EJBCA CAs, use Keyfactor Command private key retention (see Details Tab on page 387).
Principal Name	The user principal name (UPN) contained in the subject alternative name (SAN) field of the certificate, if present (e.g. username@keyexample.com).
Requester Name	The name of the requester in DOMAIN\User format.

Validation Tab

This tool will report on the certificate validity based on the criteria defining the status of an X509 chain shown in Table 2: Validation Tab Descriptions. This tab replaces the former **Validate** action from the certificate search grid. An alert symbol will show on the tab header if one or more tests have a result of *Fail*.

Tip: See Certificate Validation Errors on page 828 for assistance troubleshooting validation errors.

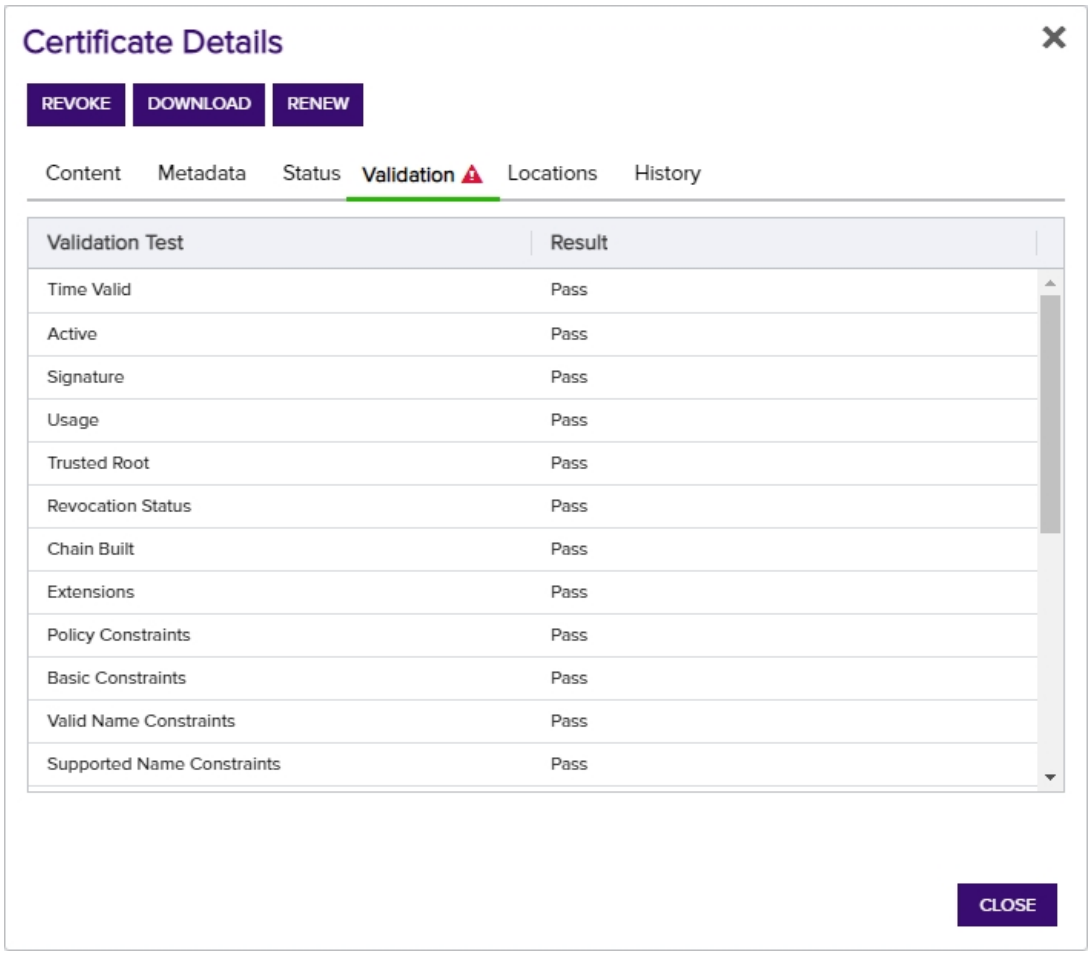


Figure 23: Certificate Details: Validation Tab

Table 2: Validation Tab Descriptions

Validation Test	Keyfactor API Equivalent[1]	Definition
Time Valid	NotTimeValid	A value of *Pass* indicates that the certificate time value is valid. A time can appear invalid (*Fail*) for a certificate that has expired.
Active	Revoked	A value of *Pass* indicates that the X509 certificate chain is valid for the certificate and contains no revoked certificates or errors.
Signature	NotSignatureValid	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid certificate signature.
Usage	NotValidForUsage	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid key usage.
Trusted Root	UntrustedRoot	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an untrusted root certificate.
Revocation Status	RevocationStatusUnknown	A value of *Pass* indicates that the revocation status can successfully be determined for the certificate. This may be the result of successful access to online certificate revocation lists (CRLs) and, if configured, authority information access (AIA) endpoints.
Chain Built	Cyclic	A value of *Pass* indicates that the certificate chain for the certificate could successfully be built.
Extensions	InvalidExtension	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid extension.
Policy	InvalidPolicyConstraints	A value of *Fail* indicates that the X509 certi-

[1]The parameter names for results returned by the Keyfactor API ***GET /Certificates/{id}/Validate*** method.

Validation Test	Keyfactor API Equivalent[1]	Definition
Constraints		ficate chain is invalid as a result of an invalid policy constraint.
Basic Constraints	InvalidBasicConstraints	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid basic constraint.
Valid Name Constraints	InvalidNameConstraints	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid name constraint.
Supported Name Constraints	HasNotSupportedNameConstraint	A value of *Fail* indicates that a name constraint for the certificate is unsupported or that the certificate has no supported name constraints.
Defined Name Constraints	HasNotDefinedNameConstraint	A value of *Fail* indicates that a name constraint for the certificate is undefined.
Permitted Name Constraints	HasNotPermittedNameConstraint	A value of *Fail* indicates that a name constraint for the certificate is impermissible.
Excluded Name Constraints	HasExcludedNameConstraint	A value of *Fail* indicates that a name constraint for the certificate has been excluded.
Full Chain	PartialChain	A value of *Pass* indicates that the certificate chain for the certificate could successfully be built up to the root certificate.
CTL Time Valid	CtlNotTimeValid	A value of *Fail* indicates that the certificate trust list (CTL) is invalid because of an invalid time value (e.g. the CTL has expired).
CTL Signature Valid	CtlNotSignatureValid	A value of *Fail* indicates that the certificate trust list (CTL) contains an invalid signature.

[1]The parameter names for results returned by the Keyfactor API ***GET /Certificates/{id}/Validate*** method.

Validation Test	Keyfactor API Equivalent[1]	Definition
CTL Usage Valid	CtlNotValidForUsage	A value of *Fail* indicates that the certificate trust list (CTL) is not valid for this use.
Strong Signature	HasWeakSignature	A value of *Pass* indicates that the certificate has been signed with a secure hashing algorithm. A value of *Fail* can indicate that a hashing algorithm of MD2 or MD5 was used for the certificate.
CRL online	OfflineRevocation	A value of *Pass* indicates that the online certificate revocation list (CRL) the chain relies on is available.
Chain Policy	NoIssuanceChainPolicy	A value of *Pass* indicates that there is either no certificate policy by design in the certificate or that if a group policy has specified that all certificates must have a certificate policy, the certificate policy exists in the certificate.
No Explicit Distrust	ExplicitDistrust	A value of *Pass* indicates that the certificate is not explicitly distrusted.
Critical Extensions	HasNotSupportedCriticalExtension	A value of *Pass* indicates that the certificate has a critical extension that is supported or has no critical extensions.

Locations Tab

If you have added the certificate to any certificate store location(s) a number will appear in the **Count** column on the corresponding **Location Type** row. Users with limited permissions will only see locations for types of certificate stores to which they have been granted permissions either globally or via certificate store containers (see Container Permissions on page 629). Click the count number for more details regarding this certificate's location. See Add to Certificate Store on page 45 for more information. The *Total Cert Store Locations* appears at the end of the list. Clicking on the total will open a dialog with the list of locations with the columns: Store Path, Store Machine, Alias, IPAddress, Port, and Agent Pool which will be populated depending on the details of the individual stores.

Note: The SSL network name is searchable with certificate search and also appears in the location details grid of the certificate details, if the certificate was found during an SSL scan.

[1]The parameter names for results returned by the Keyfactor API ***GET /Certificates/{id}/Validate*** method.

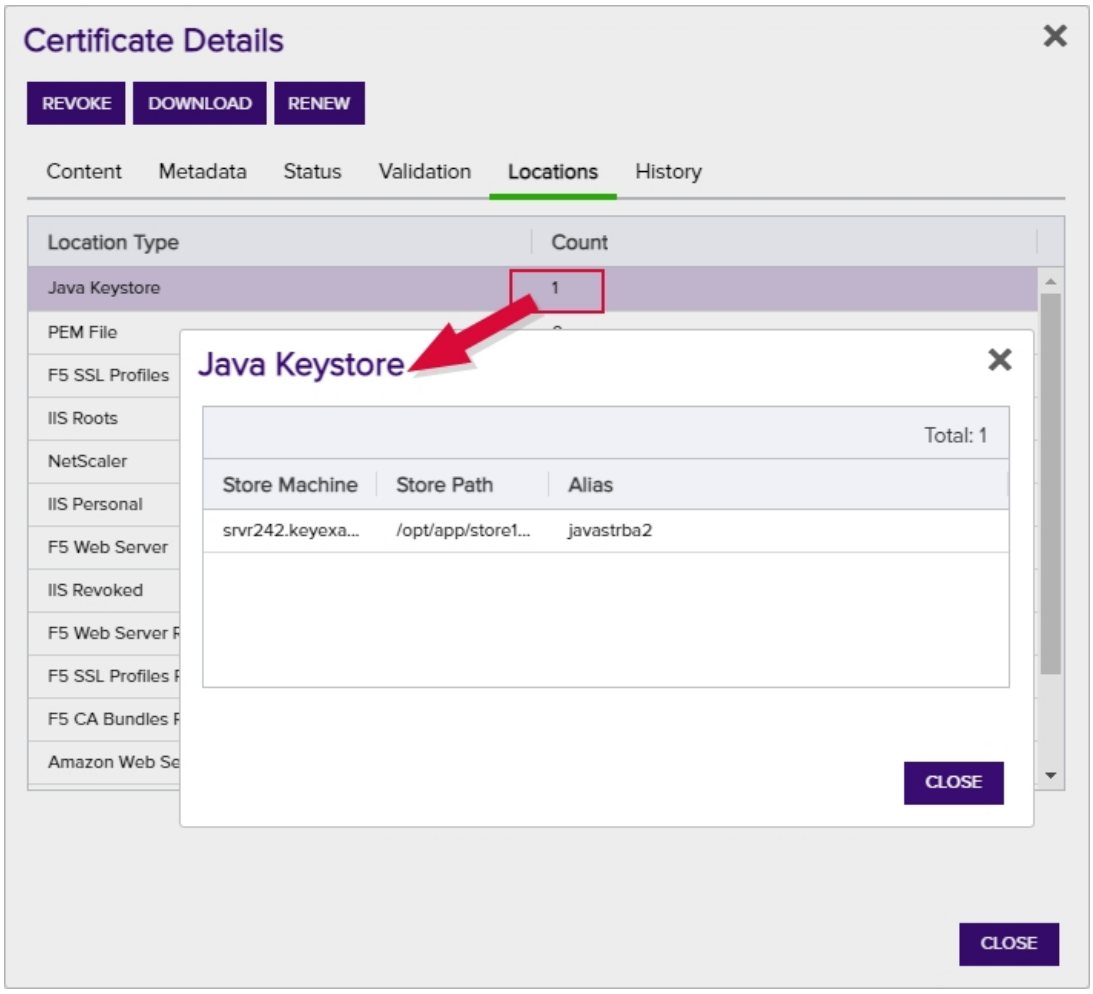


Figure 24: Location Details

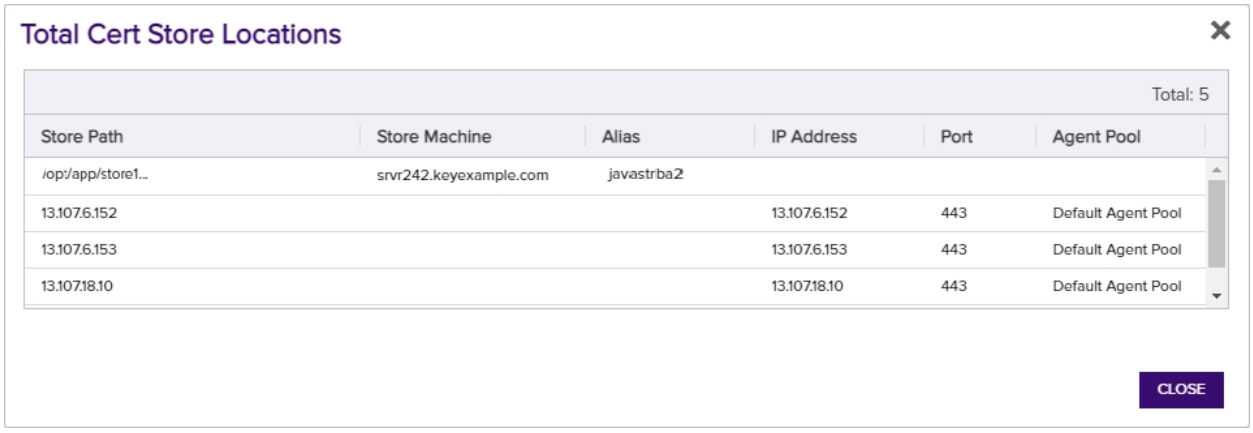


Figure 25: Total Certificate Store Location Details

History Tab

History about a certificate is recorded in the Keyfactor Command database for the following types of activities (see also Audit Log on page 716):

- Initial Import—A history entry is made on import via CA synchronization, SSL synchronization, certificate store synchronization or manual import.
- Certificate Enrollment—A history entry is made when a PFX or CSR enrollment is completed through the Keyfactor Command Management Portal. The source of the request (PFX or CSR) is indicated.
- Revocation—A history entry is made each time a certificate is revoked, so if a certificate is revoked multiple times, there will be multiple history entries.
- Key Recovery—A history entry is made each time the key for a certificate is recovered, so if the key for a given certificate is recovered multiple times, there will be multiple history entries. This type of record is generated when the private key for a certificate is downloaded from the Keyfactor Command database or when a private key is recovered from a CA using the CA's key recovery mechanism.
- Certificate Store Additions and Removals—A history entry is made each time a certificate is added to a certificate store or removed from a certificate store. These entries reference the specific certificate store type and whether the operation was an addition or removal—*Add ([store type])* and *Remove ([store type])*—and include details in the certificate history comments.
- Certificate Renewals—A history entry is made each time a certificate is renewed or reissued. The certificate renewal history record appears on both the old certificate (renewed from) and the new renewed certificate.
- Certificate Store Inventory Discoveries—A history entry is made each time an inventory of a certificate store notices that a certificate that was in a certificate store no longer is or that a new certificate has appeared in the certificate store. These entries are referenced as *Certificate Appeared ([store type])* and *Certificate Disappeared ([store type])* with details in the certificate history comments.
- SSL Endpoint Inventory Discoveries—A history entry is made each time an inventory of an SSL endpoint notices that a certificate that was at an endpoint no longer is or that a new certificate has appeared at an endpoint during a monitoring task. These entries are referenced as *Certificate Appeared (SSL Sync)* and *Certificate Disappeared (SSL Sync)* with details in the certificate history comments.
- Metadata Updated—A history entry is made each time a metadata field is updated for the certificate. The changed data will be recorded in the *Comment* field.

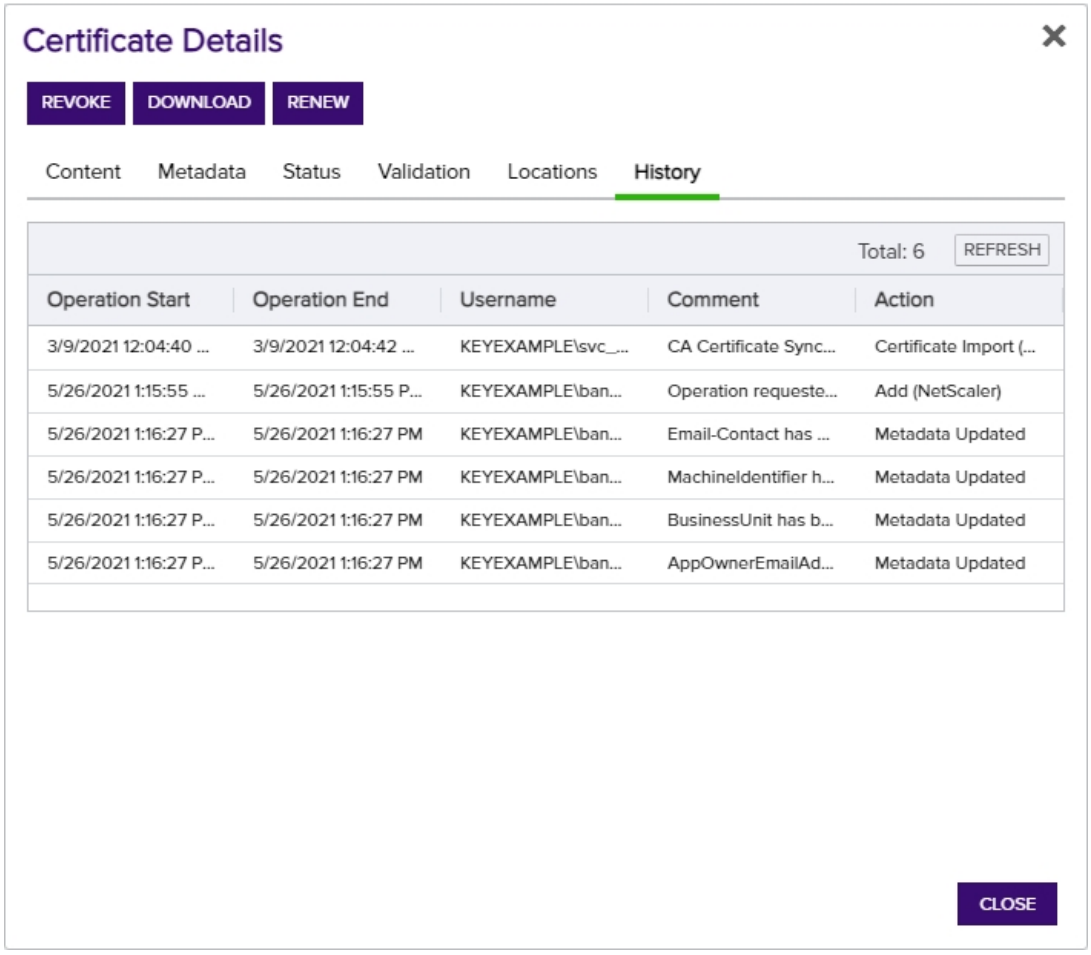


Figure 26: Certificate Operation: Certificate History Tab

Tip: Double-click a row on the History grid to see the content of that row in a more readable pop-up.

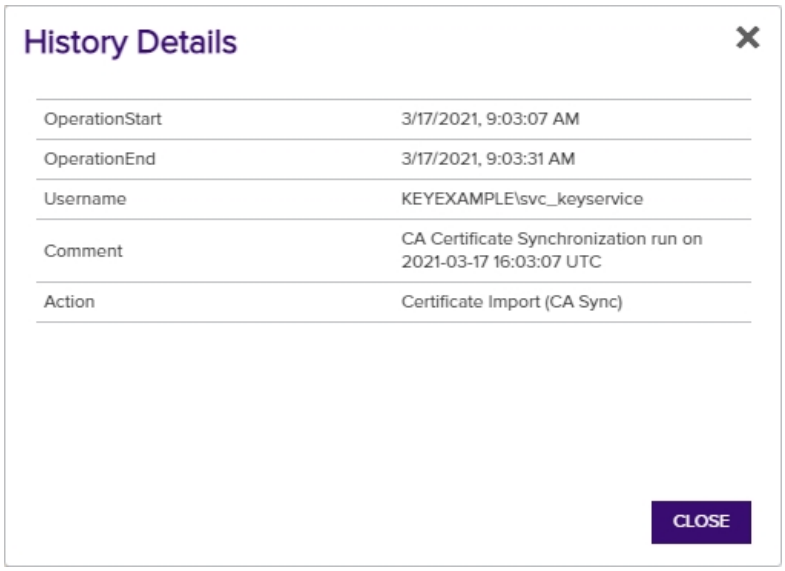


Figure 27: Certificate Operation: Certificate History Detail

2.1.3.2 Certificate Search Page

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

ArchivedKey

The certificate's archived key has been encrypted and saved to the Keyfactor Command database (true/false).

CertId

Numeric matches with the Keyfactor Command reference ID for the certificate.

CA

KeyType

The referenced certificate key type. Supported types are:

- Unknown
- RSA
- DSA
- ECC
- DH
- Ed448

Complete or partial matches with the certificate issuing CA logical name.

CertState

The certificate state; Unknown, Active, Revoked, CertificateAuthority, ParentCertificateAuthority.

CertStoreFQDN

Complete or partial matches with the fully qualified domain name of the computer hosting one or more certificate stores.

This field has an alias of *JavaKeystoreFQDN* that may be used when querying the field from the Keyfactor API.

CertStorePath

Complete or partial matches on the full path to a certificate store—e.g. /opt/application/mystore.jks or c:\program files\application\mystore.jks.

This field has an alias of *JavaKeystorePath* that may be used when querying the field from the Keyfactor API.

CertStoreContainer

Certificate is in a certificate store that is included in the container criteria indicated.

CN

Complete or partial matches with the certificate common name.

This field has an alias of *IssuedCN* that may be used when querying the field from the Keyfactor API.

DN

Complete or partial matches with the certificate distinguished name.

This field has an alias of *IssuedDN* that may be used when querying the field from the Keyfactor API.

ExpirationDate

Certificate expiration before, after, or on a

- Ed25519
- Dilithium2
- Dilithium3
- Dilithium5

KeyUsage

Certificate includes or doesn't include (or is null or not null for) the referenced key usage. Supported key usages are:

- CRLSign
- DataEncipherment
- DecipherOnly
- DigitalSignature
- EncipherOnly
- KeyAgreement
- KeyCertSign
- KeyEncipherment
- NonRepudiation

NetBIOSPrincipal

Complete or partial matches with the certificate principal name in NetBIOS format (DOMAIN\username). Supports the %ME% token (see Advanced Searches on page 40).

This field has an alias of *PrincipalName* that may be used when querying the field from the Keyfactor API.

NetBIOSRequester

Complete or partial matches with the certificate requester's name in NetBIOS format (DOMAIN\username). Supports the %ME% token (see Advanced Searches on page 40).

This field has an alias of *RequesterName* that may be used when querying the field from the Keyfactor API.

OU

Complete or partial matches with the certificate organizational unit.

specified date. Supports the %TODAY% token (see Advanced Searches on page 40). Be sure to check the *Include Expired* checkbox to view expired certificates.

This field has an alias of *NotAfter* that may be used when querying the field from the Keyfactor API.

EKU

Complete or partial matches with the certificate template OID.

EKUName

Complete or partial matches with the certificate template Name.

HasPrivateKey

Certificate private key encrypted and stored in the Keyfactor Command database (true/false).

ImportDate

The certificate imported to Keyfactor Command before, after, or on a specified date.

IssuedDate

Certificate issuance before, after, or on a specified date. Supports the %TODAY% token (see Advanced Searches on page 40).

This field has aliases of *NotBefore* and *EffectiveDate* that may be used when querying the field from the Keyfactor API.

IssuerDN

Complete or partial matches with the certificate issuer's distinguished name.

KeyfactorRequestId

Numeric matches with the Keyfactor Command reference ID for the certificate request.

KeySize

Complete or partial matches with the certificate

PublicKey

Exact matches with the certificate public key in hexadecimal or base64 format.

RevocationDate

Certificate revocation before, after, or on a specified date, or is null or not null. Be sure to check the *Include Revoked* checkbox to view revoked certificates. Supports the %TODAY% token (see Advanced Searches on page 40).

This field has an alias of *RevocationEffDate* that may be used when querying the field from the Keyfactor API.

Revoker

Complete or partial matches with the name of the user (DOMAIN\username format) who revoked the certificate. Be sure to check the *Include Revoked* checkbox to view revoked certificates.

RFC2818Compliant

Certificate is compliant with RFC 2818 (contains a DNS SAN) (true/false).

SelfSigned

Certificate is self-signed (true/false).

SerialNumber

Complete, or starts/ends with, or null/not null matches with the certificate serial number.

SigningAlgorithm

Complete or partial matches with the certificate signing algorithm.

SSLDNSName

Complete or partial matches with the DNS name resolved for an SSL endpoint.

SSLIPAddress

key size.

This field has an alias of *KeySizeInBits* that may be used when querying the field from the Keyfactor API.

Complete, or starts/ends with, or null/not null matches with the IP address defined for an SSL endpoint.

This field has an alias of *SslHostName* that may be used when querying the field from the Keyfactor API.

SSLNetworkName

Complete, or starts/ends with, or null/not null matches with the network name under which an SSL endpoint was found.

SSLPort

Complete or partial numeric matches with the port number defined for an SSL endpoint.

SAN

Complete or partial matches with the certificate subject alternate name(s).

TemplateDisplayName

Complete or partial matches with the certificate template display name.

This field has an alias of *TemplateName* that may be used when querying the field from the Keyfactor API.

TemplateShortName

Complete or partial matches with the certificate template name.

Thumbprint

Complete or partial matches with the certificate thumbprint value.

You can also do queries based on user-defined metadata fields (see Certificate Metadata on page 710).

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the

equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options. The results grid includes these fields:

Issued DN

The distinguished name of the certificate subject.

Import Date

The date the certificate was imported to Keyfactor Command. This field will auto populate on any new imports/enrollments of certificates. On an upgrade, this field will be populated in existing certificates from the certificate operation history.

Principal Name

The identity that the certificate represents. The principal name field is populated during certificate synchronization by the user principal name (UPN) extracted from Active Directory if there is a principal name in the certificate subject alternative name (SAN).

Requester

The user or entity that requested the certificate.

Effective Date

The date the certificate was issued or became active.

Expiration Date

The date the certificate expires.

Issued CN

The common name of the certificate subject.

Issuer DN

The distinguished name of the certificate issuer.

Certificate Template

The short name of the template used to issue the certificate.

Locations

The server(s), if any, that the certificate is hosted on (e.g. for SSL certificates). If the certificate is found on multiple servers, this field will show the number of servers on which it was found and the location type (e.g. *4 SSL* or *6 JKS*). The specific server names can be found in the certificate details.

Key Type

The key type of the certificate.

Key Size

The key size of the certificate.

Certificate State

The certificate state options are:

- Unknown (0)
- Active (1)
- Revoked (2)
- Failed (4)
- Pending (5)
- Certificate Authority (6)
- Parent Certificate Authority (7)

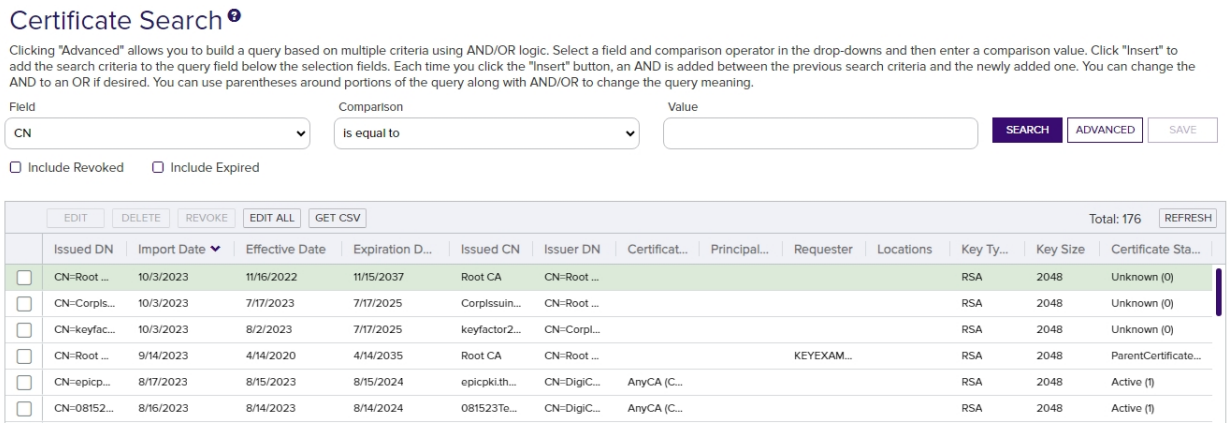


Figure 28: Certificate Search

The search results can be sorted by clicking on a column header in the results grid for every column (except Certificate Locations, Key Type, and Certificate State). Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and

dragging the header of the column you wish to move. The column widths may be adjusted by click-holding and dragging the line separating two column headers.

You can click the **Include Revoked** and/or **Include Expired** buttons at the top of the results grid to toggle inclusion of revoked or expired certificates in the results. By default they are excluded.

The rest of the buttons at the top of the display grid are used to interact with the certificates displayed in the results grid. Some buttons are grayed out until you click on a grid row. Other certificate functions are available on the right-click menu. To open the right-click menu, highlight a row in the results grid and right-click. You can also double-click a certificate row in the results grid to open the Certificate Details (see Certificate Details on page 19).

To select a single row in the grid, click to highlight it and then select an operation from either the top of the grid or the right-click menu. Some of the certificate operations support action on multiple certificates at once. To select multiple rows, hold down the CTRL key and click each row on which you would like to perform an operation, or tick the check box next to the row. Then select an operation from the top of the grid. The right-click menu supports limited operations on the multiple certificates.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

Note: Multiple *OR* queries can be slow due to the nature of the query not being indexed and potentially requiring multiple queries of the database. To mitigate this, we suggest you create a collection for the subset of certificates, using the *OR* statement as needed, then perform a

search starting with that collection and adding any additional conditions using advanced search from the search page. See Saving Search Criteria as a Collection on the next page.

In addition to the options available in the query builder, three special values can be used in selected searches by typing them in directly:

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).

  Example: Create a certificate search of IssuedDate -ge "%TODAY-7%" and save it as a collection called *Certificates Issued in the Last Week*. Create another certificate search of ExpirationDate -lt "%TODAY+60%" and save it as a collection called *Certificates Expiring in the Next 60 Days*. This allows you to have saved collections containing a comparison date without having to update the date in the collection.

- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).

  Example: Create a certificate search of NetBIOSRequester -contains "%ME%" and save it as a collection. Multiple users can now use this same collection to search for all the certificates on which they were the requester in the current domain.

  Note: Certificate collections saved using the %ME% value are *not* supported for use in reports or on the dashboard.

- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

  Example: Create a certificate search of NetBIOSRequester -contains "%ME-AN%" and save it as a collection. Multiple users can now use this same collection to search for all the certificates on which they were the requester, regardless of domain.

  Note: Certificate collections saved using the %ME-AN% value are *not* supported for use in reports or on the dashboard.

Important: The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

To build a deep link with your search criteria, begin with the following URL (where KEYFACTOR_SERVER_FQDN is the FQDN of your Keyfactor Command administration server):

> https://KEYFACTOR_SERVER_FQDN/keyfactorportal/CertificateCollection/Query?query=YOUR_URL_ENCODED_QUERY

Your Management Portal may have been configured to use HTTP rather than HTTPS.

Replace YOUR_URL_ENCODED_QUERY with your search criteria as built using the advanced search. The search criteria needs to be URL encoded, so, for example, spaces need to be replaced with %20 and quotation marks with %22. However, many modern browsers will automatically do this for you. A deep link using part of the example search shown above would look something like this without URL encoding:

> https://keyfactor.keyexample.com/keyfactorportal/CertificateCollection/Query?query=CN -contains ”appsrvr”

And with URL encoding, like this:

> https://keyfactor.keyexample.com/keyfactorportal/CertificateCollection/Query?query=CN%20-contains%20%22appsrvr%22

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Saving Search Criteria as a Collection

To save your search criteria as a certificate collection:

1. Click the **Save** button.

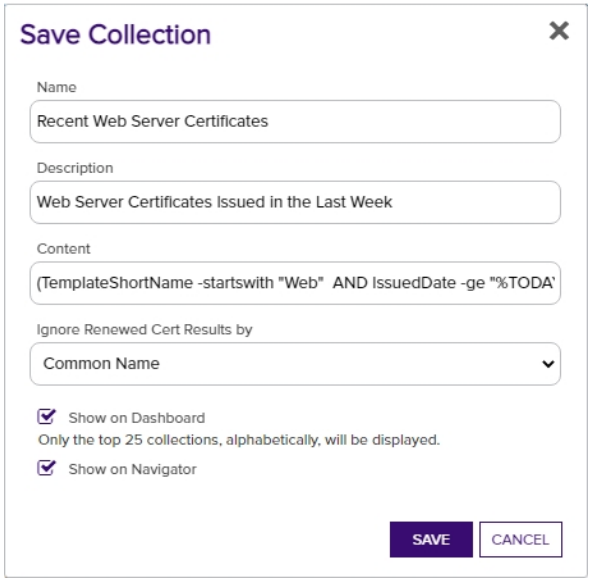


Figure 29: Save Certificate Collection

2. In the Save Certificate Search dialog, enter a name for the certificate collection. This name appears at the top of the page for this collection and can be configured to appear on the Management Portal menu under Certificates. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports and dashboards). Because it can appear on the menu and in selection drop-downs, the name should be fairly short.
3. Enter a description for the collection. This description appears as a subtitle below the collection name on the page for this collection and can be more detailed than the collection name.
4. Select a setting in the *Ignore renewed certificate results by* dropdown. The *Ignore* dropdown applies to processing reports or expiration alerts and contains these options:

None

Do not eliminate duplicate certificates when processing reports or expiration alerts based on this certificate collection.

Common Name

Eliminate duplicate certificates based on the common name in the certificate when processing reports or expiration alerts. Certificates will be excluded from reports and expiration alerts if they share the same common name and enhanced key usage (EKU—e.g. Client Authentication). The certificate with the most recent issued date and the given common name and EKU will be included in the report or expiration alert.

Distinguished Name

Eliminate duplicate certificates based on the distinguished name in the certificate when processing reports or expiration alerts. Certificates will be excluded from reports and expiration alerts if they share the same distinguished name and EKU. The certificate with the most recent issued date and the given distinguished name and EKU will be included in the report or expiration alert.

Principal Name

Eliminate duplicate certificates based on the principal name in the certificate status data stored in the Keyfactor Command database for the certificate when processing reports or expiration alerts. The principal name is added to the certificate status data for the certificate during certificate synchronization if the certificate SAN contains a *user principal name* or *NT principal name*. Certificates will be excluded from reports and expiration alerts if they share the same principal name and EKU. The certificate with the most recent issued date and the given principal name and EKU will be included in the report or expiration alert.

Keyfactor Renewal

Eliminate duplicate certificates based on certificates that have been renewed through Keyfactor Command.

Note: Regardless of the selection you make in the Ignore option, all certificates will appear in the search results grid. Duplicate certificates are not excluded on this page. When processing reports or expiration alerts based on this certificate collection, only certificates that share all the EKUs (e.g. Client Authentication and Server Authentication) as well as the same CN, DN or UPN will be eliminated as duplicates. If a certificate has more than one EKU and at least one EKU does not match an otherwise similar certificate with matching CN, DN or UPN, it will not be eliminated on reports or expiration alerts.

5. Check the **Show on Dashboard** box to include the results from this collection on the *Collection* dashboard (see Dashboard: Collections on page 12). You will not be able to change this setting once the collection is saved. If you need to change it, you would need to edit the collection and re-save it.

   Note: The collections dashboard widget will only display the first 25 collections alphabetically. A brief warning message explaining this will be shown on the collections save dialog when the **Show on Dashboard** box is checked.

6. Check the **Show in Navigator** box to include the collection on the Management Portal menu (on the *Certificates* top-level menu dropdown).

7. Click **Save** to save the collection. The search results will display immediately. If you didn't select the **Show in Navigator** option, you can find the collection again on the Certificate Collection

Management page, accessed by navigating to *Certificates > Collection Manager* from the Management Portal.

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

2.1.3.3 Certificate Operations

Most common certificate operations (except enrollment) are available on the Certificate Search grid. The actions available on the grid header include: **Edit** (users with read-only permissions will see **Display** instead), **Delete**, **Revoke**, **Edit All**, **Revoke All**, **Delete All** (for collections only), and **Get CSV**. Secondary operations are shown on the context menu, accessed by right-clicking on a selected row on the Certificate Search grid. The context menu includes **Edit** (or **Display**), **Delete**, **Delete Private Key**, **Revoke**, **Download**, **Add to Certificate Store**, **Remove from Certificate Store**, **Renew**, and **Identity Audit**. There is also an operation to place a hold, or remove a hold, on a certificate, which is available from the Revoke operation through the Revocation Reason: Certificate Hold/Remove From Hold. When selecting multiple rows, only the operations Edit, Delete, Revoke and Delete Private Key (only if the private key is stored in the database) are enabled on the grid header and the context menu. For the edit commands, the only details that can be edited are the metadata fields.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

Full descriptions of the available certificate operations are below.

Add to Certificate Store

Before adding a certificate to a certificate store in Keyfactor Command, you must approve an orchestrator to handle the store and create a record for the store in Keyfactor Command. See Orchestrator Management on page 496 and Certificate Store Operations on page 413.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Private Key > Read
Certificate Stores > Read
Certificate Stores > Schedule

Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. See Certificate Collection Permissions on page 627 and Container Permissions on page 629 for more information about global vs collection and container permissions.

Note: Certificates cannot be added to stores that require private keys from this interface unless the selected certificate contains a private key stored in the database. If the selected certificate does not contain a stored private key, stores that require a private key will not appear on the Select Certificate Store Locations dialog.

To add a certificate to a certificate store:

1. Highlight the row in the results grid and right-click.
2. Choose **Add to a Certificate Store** from the right-click menu.
3. When you select the Add to Certificate Store option the *Select Certificate Store Locations* dialog opens. When you select the certificate stores to which you want to deploy your certificate and click **Include**, the *Add to Certificate Stores* dialog appears BEHIND the *Select Certificate Store Locations* dialog, holding your selection and leaving the *Select Certificate Store Locations* dialog open for you to continue selecting locations. The final list of selections will only be accessible once you close the *Select Certificate Store Locations* dialog using the **Include and Close** button.

Select Certificate Store Locations

The *Select Certificate Store Locations* dialog allows you to run queries against your certificate store list to select which store(s) to deploy a selected certificate to. **Check** the box next to each certificate store location to which you want to distribute the certificate.

Note: Only compatible certificate stores and only stores in containers to which you have permissions are shown on the grid.

Tip: You may change the search results by using the search fields at the top of the dialog. All of the Keyfactor Command grid search features are available to assist your search. See Using the Certificate Store Search Feature on page 410 for more information on the available search fields. The default search criteria is *AgentAvailable is equal to True*.

The actions on the *Select Certificate Store Locations* dialog are:

- **Include**

  Click this to add the selected certificate store(s) to your certificate selection and leave the search dialog open for further searches.

- **Include and Close**

  Click this to close the search dialog and add the selected certificate store(s) to your certificate selection, which will then be displayed and ready for updates as per the instructions in *Add to Certificate Stores*.

- **Close**

  Click this to cancel the operation and return to the main page with no certificate stores selected.

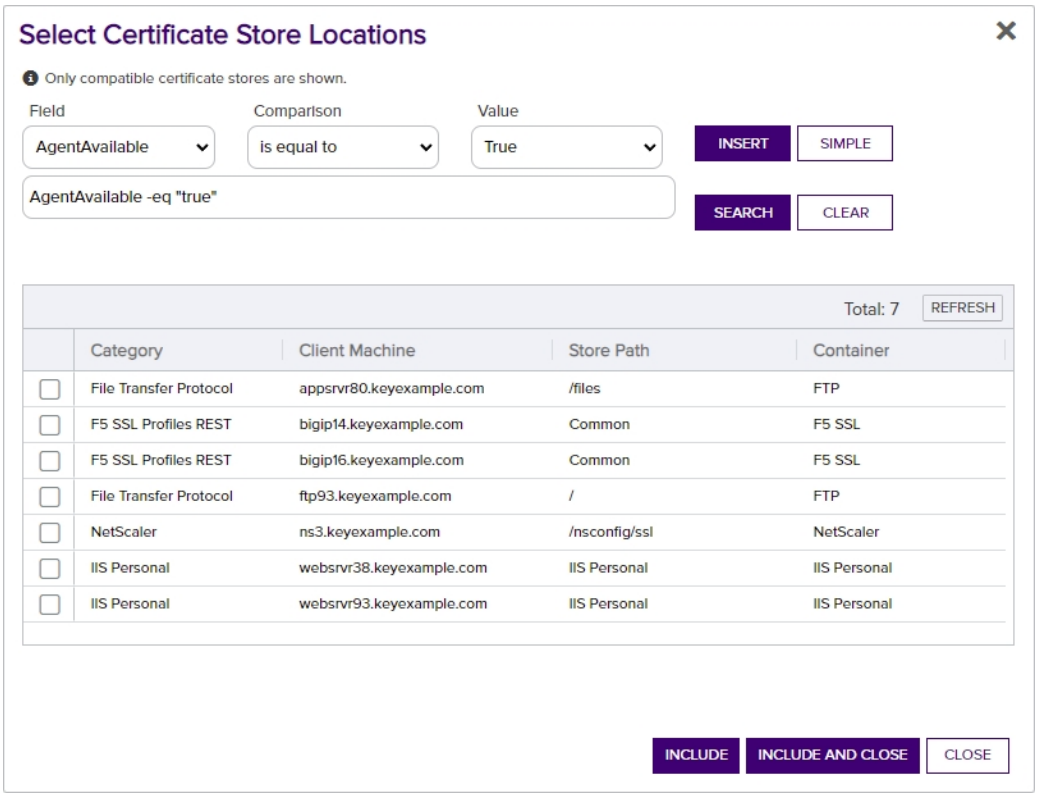


Figure 30: Select Certificate Store Locations Dialog

Add to Certificate Stores

The *Add to Certificate Stores* page appears once you select at least one certificate store to distribute your certificate to. It includes a grid section with a series of tabs that display a tab for each type of certificate store selected with a list of the selected stores under each tab. The header section of the dialog shows global options that apply to the add job as a whole:

- **Include Certificate Stores**

  You may return to the *Select Certificate Store Locations* dialog by clicking **Include Certificate Stores** above the grid. The current selections will be retained.

- **Schedule when to run the job for the certificate store**

  In the **Schedule** dropdown, select a time at which the job to add the certificate to the stores should run. The choices are *Immediate* or *Exactly Once* at a specified date and time. If you choose *Exactly Once*, enter the date and time for the job. A job scheduled for *Immediate* running will run within a few minutes of saving the operation. The default is *Immediate*.

- **Include Private Key on Certificate Stores when the Private Key is optional**

  Check the **Include Private Key** box if you want to deliver the private key of the certificate to any selected certificate stores that do not require a private key (e.g. Java keystores). This option only appears for certificates that have a private key available for distribution.

Click **Remove** at the top of the grid to remove the selected certificate store from the page. The certificate will not be added to the store.

For each selected certificate store you can apply the following actions:

- **Overwrite**

  Check **Overwrite** below the grid to overwrite any existing certificate in the same location and with the same name or alias for the selected certificate store type.

- **Alias**

  Add an **Alias** below the grid, if applicable, for the certificate store type. See the **Information Required by Certificate Store** section, below, for more information.

  Note: The tab heading of the certificate location will display an alert if an alias is required for the location. If *Supports Custom Alias* is set to **Forbidden** on the certificate store type, the **Alias** field will not display unless *Overwrite* is checked on this page.

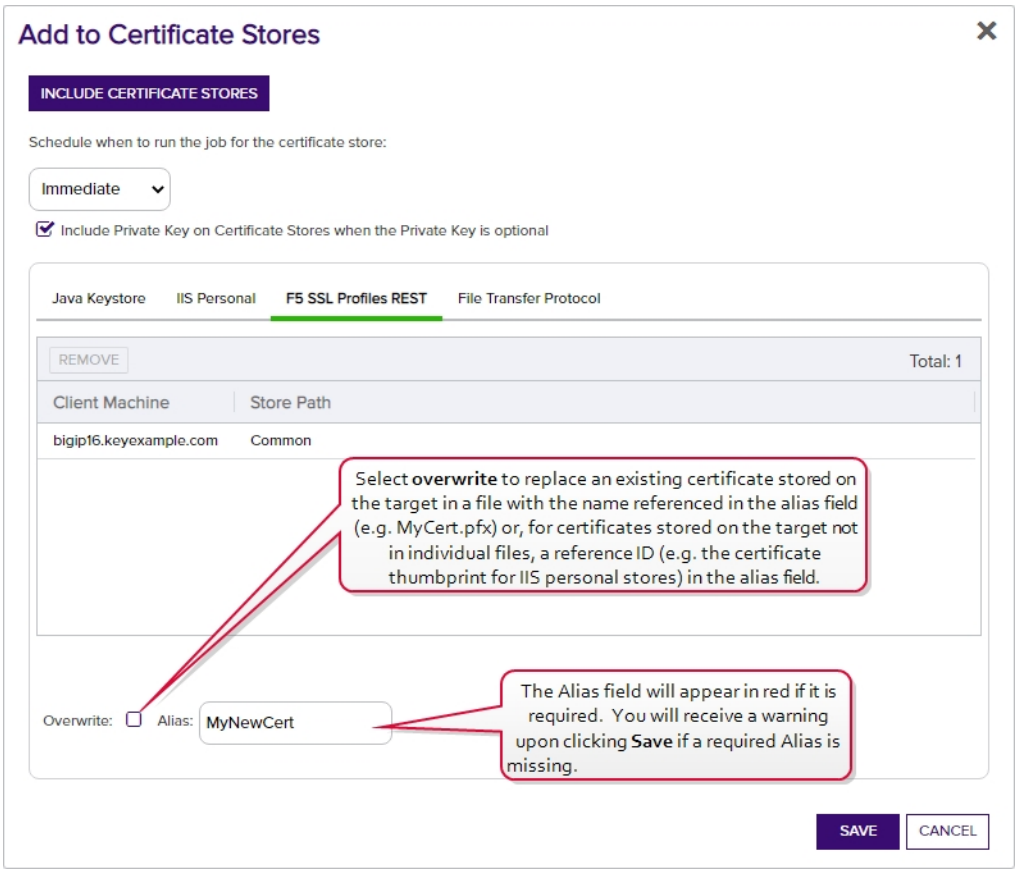


Figure 31: Add Certificate—Install into Certificate Locations

Figure 32: Alias Required Alert on Save

Information Required by Certificate Stores

Each type of certificate store has different requirements for providing an alias or other additional information.

The certificate store type fields that are relevant to certificate store use are:

- **Supports Entry Password**

  If your certificate store type has this enabled, you will have the option to enter a password for the certificate entry in the certificate store on the addition of an entry into the certificate store.

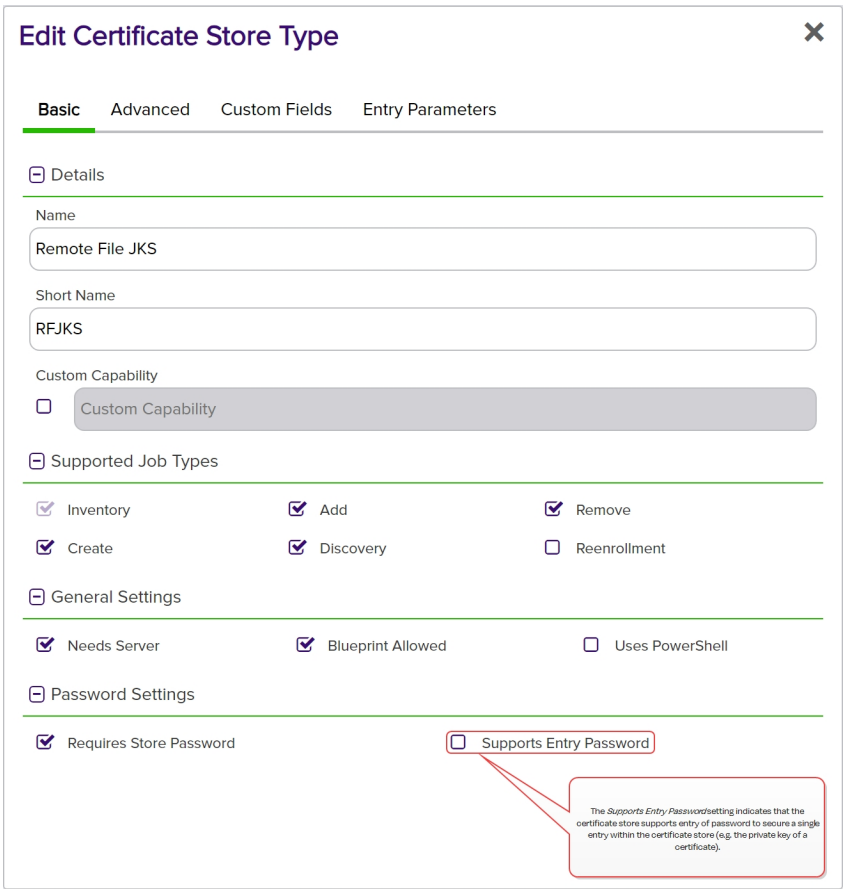


Figure 33: Certificate Store Type Configuration: Basic Tab

- **Supports Custom Alias**

  A value of *Required* indicates that a custom alias will be required when a certificate is added to a certificate store. *Optional* indicates an alias can be associated with the entry if desired. If your certificate store type sets this to *Forbidden*, the *Alias* field will not display when adding a certificate to a certificate store unless *Overwrite* is checked on the add page. In this case, you're not associating an alias with the certificate you're adding to the store but rather specifying the alias of the certificate already in the store that you wish to replace (in function) with the new certificate you're adding.

  The format of custom alias values varies depending on the certificate store type. In many cases, the alias is the thumbprint of the certificate. In some cases, it's the file name of the certificate file or a custom alias provided at the time the certificate was added to the certificate store. For instance:

  - For an Amazon Web Services (AWS) store, the alias is the internal ID assigned by Amazon (the Amazon resource number or ARN). Provide the entire contents of the

Alias/IP from this field when entering an alias for overwrite. For example:

arn:aws:acm:us-west-2:220531701668:certificate/88e5dcfb-a70b-4636-a8ab-e85e8ad88780

- For F5 stores using the Keyfactor custom-built F5 Certificate Store Manager extension (see Installing Custom-Built Extensions on page 2940), the alias is the file name used to store the file in the device file system, minus the extension (e.g. use alias MyFile for a file named MyFile.crt). Aliases should be entered without spaces. Note that certificate names are case sensitive.

Note: Keyfactor Command will automatically strip out any spaces between the octets in thumbprints in the alias field, so it does not matter whether you enter the thumbprint with or without spaces.

Tip: When adding a certificate to a certificate store, you have the option to overwrite an existing certificate with the current certificate. If you choose this option, you will need to provide the alias of the certificate you wish to overwrite. Find the alias values by navigating to *Management Portal > Certificates > Certificate Search*. Select the certificate you wish to overwrite and double-click, or click **Edit**, from the grid header or right-click menu. Choose the **Locations** tab and double-click on the Location Type (this must have a number other than zero in the *Count* column) to open the details dialog. The *Alias* field holds the information that may be required for an overwrite.

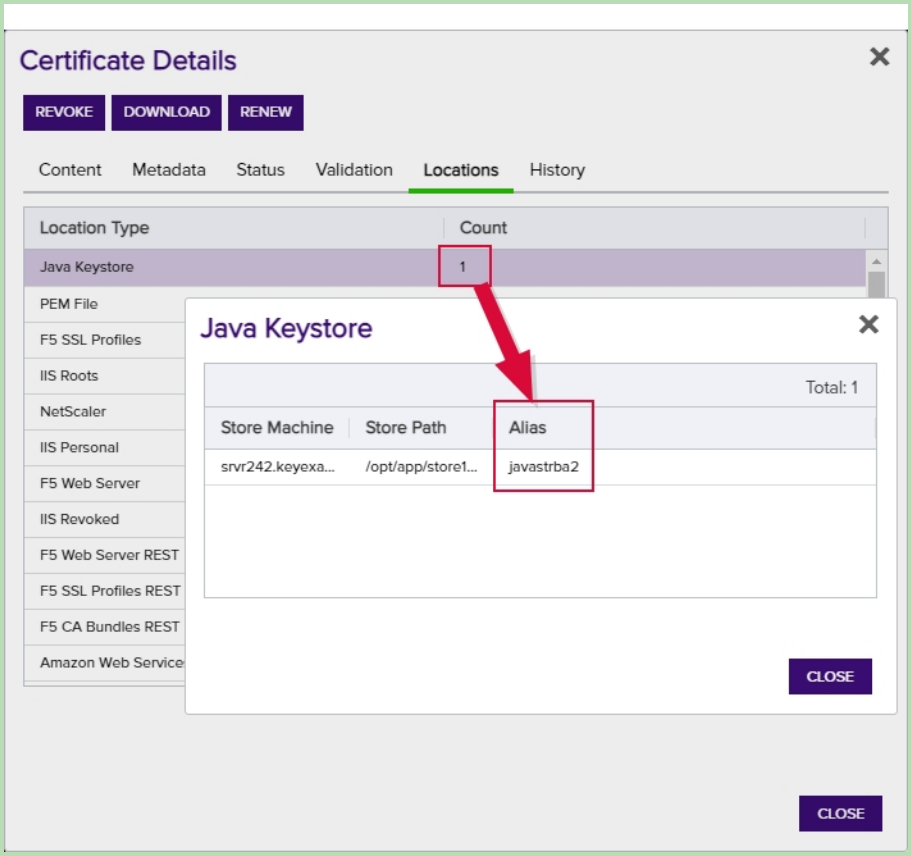


Figure 34: Example: Certificate Location Details for a JKS Location

- **Private Key Handling**

  When adding a certificate to a certificate store, if you select a certificate that does not have an associated private key, certificate stores with this option set to *Required* will not appear as available stores to which the certificate can be added. If this option is set to *Forbidden* and the selected certificate has a private key, the private key will be ignored and only the public key will be delivered to the target.

Note: Private keys are always available in PFX Enrollment.

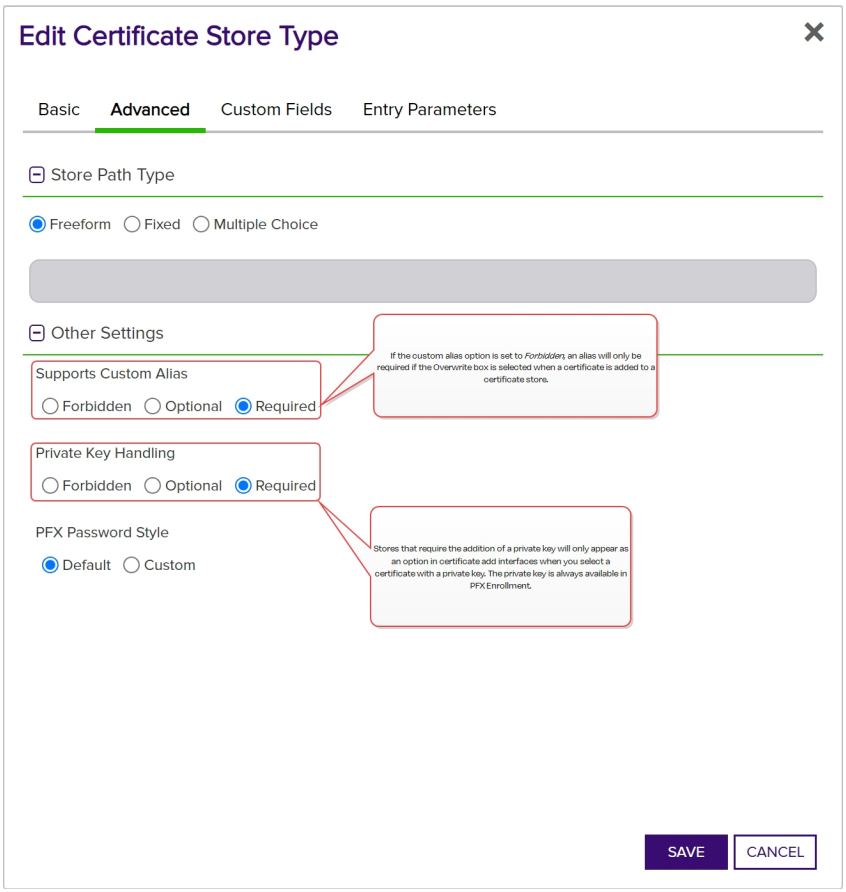


Figure 35: Certificate Store Type Configuration: Advanced Tab

- **Entry Parameters**

  Not all certificate store types will have entry parameters. The ones shown in Figure 36: Certificate Store Type Configuration: Entry Parameters Tab are for the custom *Windows Certificate* type for the Keyfactor custom-built IIS Certificate Store Manager extension (see Installing Custom-Built Extensions on page 2940).

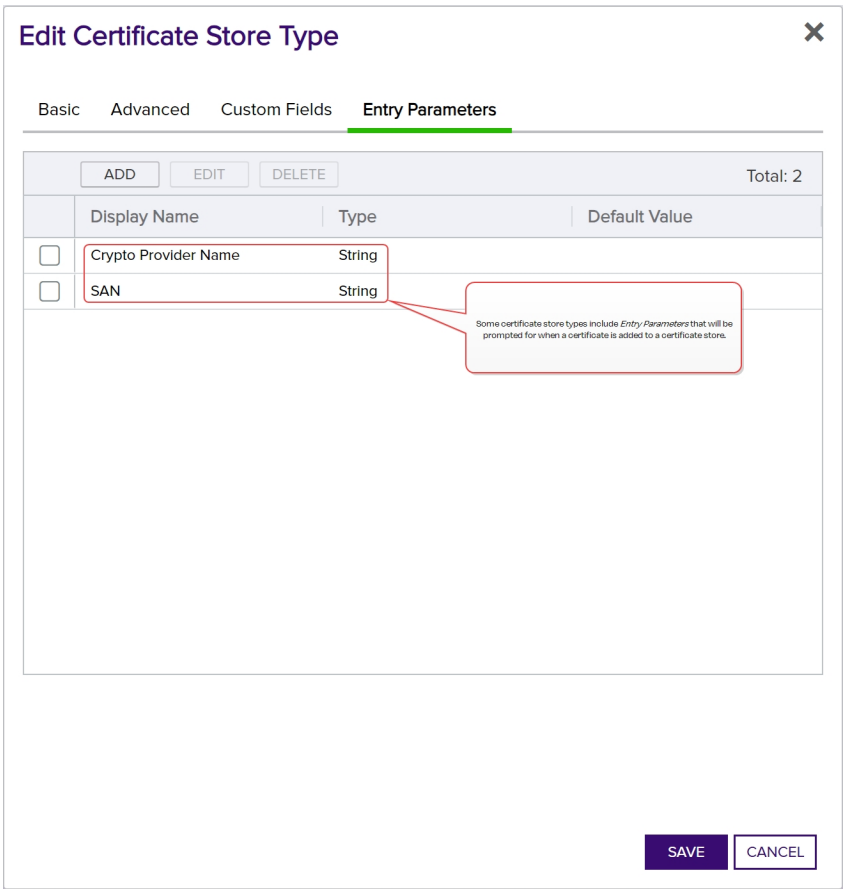


Figure 36: Certificate Store Type Configuration: Entry Parameters Tab

4. Click **Save** to submit the certificate store additions.

Note: When you save this job, a new management job will be added to the orchestrator jobs list. In addition a one-time inventory job will be added immediately following the management job to update Keyfactor Command with the changes to the certificate store. The one-time inventory job does not appear in the orchestrator jobs list in the Management Portal.

Delete

Select one or more certificates in the results grid and then click **Delete** at the top of the grid or **Delete** in the right-click menu to remove the selected certificate(s) from the Keyfactor Command database. If the selected certificates have associated private keys stored in the database, these private keys are also removed. The certificates will be returned to the Keyfactor Command database on the next full synchronization if synchronization for the certificate source (certificate authority, SSL endpoint, etc.) is still configured. Certificate history and private keys do not return when certificates re-synchronize.

Whenever a certificate is deleted that is a part of a certificate renewal chain. The certificates on either end of the deleted cert(s) will have their certificate histories updated to show that either a certificate before or after the certificate was deleted in the renewal chain of that certificate.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Delete

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Delete All

This option is available only in saved collections, not in standard certificate searches. Click the **Delete All** action button at the top of the collection grid. The button appears active only if no certificates are selected on the grid. A large deletion may take several minutes to complete. The certificates will be returned to the Keyfactor Command database on the next full synchronization if synchronization for the certificate source (certificate authority, SSL endpoint, etc.) is still configured.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Delete

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Delete Private Key

Click the **Delete Private Key** in the right-click menu to remove the private key of the selected certificate(s) from the Keyfactor Command database. This option is only available if the private key is stored in the database.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Delete

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Download

Click **Downland** in the right-click menu to download the selected certificate to the local computer with or without a private key. Only one certificate may be downloaded at a time.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Delete
To download a certificate without a private key.

Certificates > Collections > Read
Certificates > Collections > Delete
Certificates > Collections > Private Key > Read
For users who will be downloading certificates with private keys.

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Note: The Recover option that was found in previous versions of Keyfactor Command is now part of the Download option.

You will be able to download a certificate including its private key if one of the following is true:

- The certificate has been stored in the Keyfactor Command database with its private key.
- The certificate was issued using a template that had key archival enabled, issued from a Microsoft CA that has a valid Key Recovery Agent certificate, and that Key Recovery Agent certificate is configured on the Keyfactor Command server.

Important: In order to successfully download certificates and retrieve their associated private keys using Microsoft key recovery, the service account under which the Keyfactor Command application pool is running must be granted "Issue and Manage Certificates" permission to the CA database as per Create Groups to Control Access to Keyfactor Command Features on page 2762, or, if delegation is configured for the CA, the user executing the download must have these permissions.

In order to support key recovery within Keyfactor Command, you need to import at least one Key Recovery Agent certificate with a private key into the Keyfactor Command application pool user's personal certificate store on each Management Portal server. See Configuring Key Recovery for Keyfactor Command on page 789.

Tip: CA-level key recovery is supported for Microsoft CAs to allow recovery of private keys for certificates enrolled outside of Keyfactor Command. CA-level key archiving is

not supported for enrollments done through Keyfactor Command. CA-level key recovery is not supported for EJBCA CAs. For enrollments done through Keyfactor Command for either Microsoft or EJBCA CAs, use Keyfactor Command private key retention (see Details Tab on page 387).

Note: Downloading of the private key is logged and reflected on the History tab of the certificate details (see History Tab on page 32).

To download a certificate that has the private key stored in the Keyfactor Command database:

1. Highlight the row in the results grid and right-click.
2. Choose **Download** from the right-click menu, or the action button on the Certificate Details dialog.

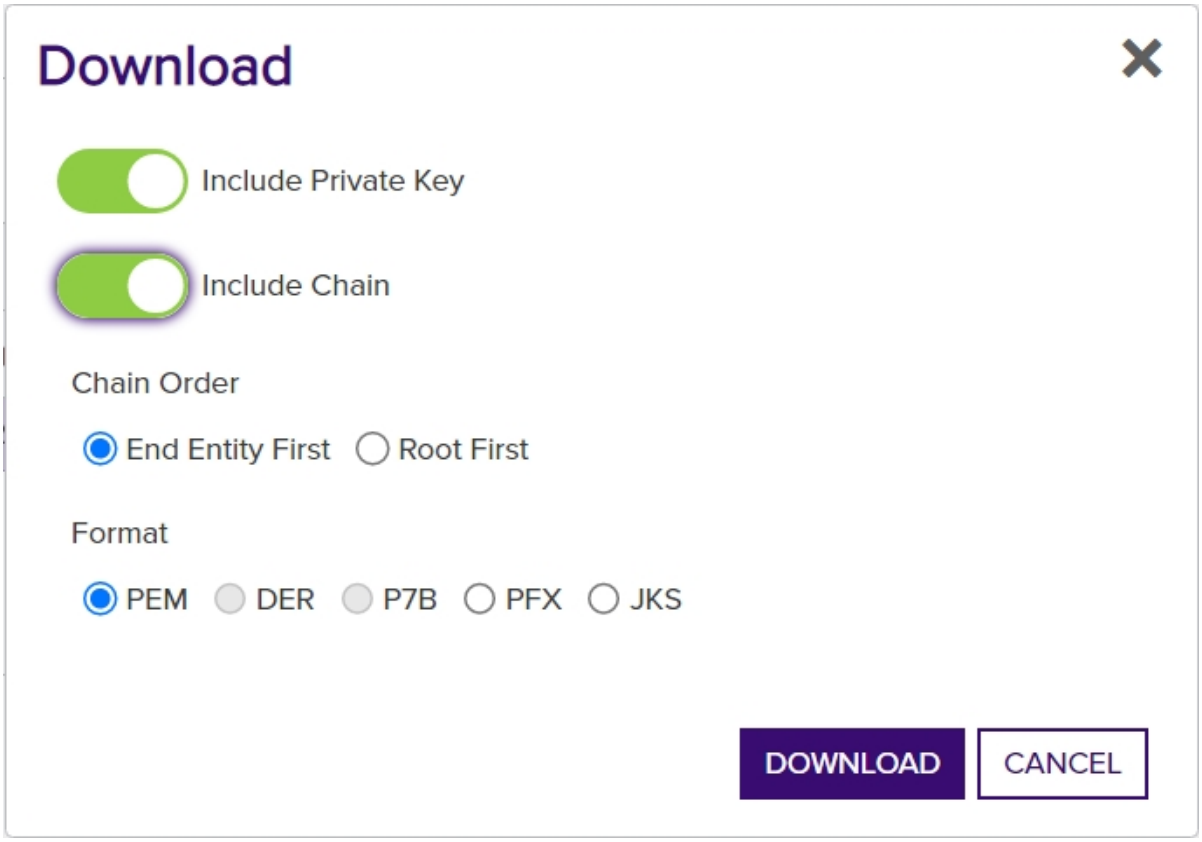


Figure 37: Certificate Operation: Download Certificate with Private Key

3. In the Download dialog, select the **Include Private Key** option to include the private key of the certificate in the download. The Include Private Key option is supported for PEM, PFX and JKS

outputs.

Note: If you choose **Include Private Key**, if you do not have the *Allow Custom Password* application setting configured to *True* (see Application Settings: Enrollment Tab on page 609), *after* you click **Download** (step 7, below), a PFX/PEM/JKS Password dialog will pop-up with the one-time password and action buttons to **Copy Password** or **Close** the pop-up. Clicking **Copy Password** will copy the password to the clipboard. As a security measure, the dialogue will close after 2 minutes. To secure the downloaded file, you will need this password in order to access the PFX, PEM, or JKS file generated by the download. Click **Close** to close the PFX/PEM/JKS Password dialog once you have copied the password.

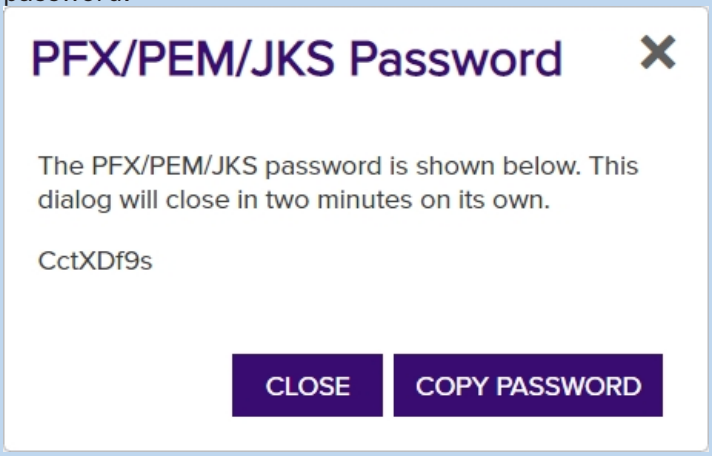


Figure 38: Certificate Operation: Password for Certificate with Private Key

Important: The randomly generated password cannot be regenerated, so it must be copied prior to closing the dialog.

4. Select **Include Chain** to include the certificate chain (root and intermediate certificates) in the download, if required.

5. If you selected Include Chain, select a **Chain Order** for the certificates in the resulting output file—either **End Entity First** (at the beginning of the file) or **Root First**. Chain Order is supported for PEM and P7B outputs. PFX output always includes the end entity certificate first.

6. Chose an encoding format (options include: DER, P7B, PEM, ZIP PEM, JKS, PFX).

Note: Selecting the *Include Private Key* and *Include Chain* options changes which formats are available.

7. If enabled, in the Password section of the page, check the **Custom Password** box and enter and confirm a custom password to use in securing the downloaded file. This section only appears if *Include Private Key* is toggled on and if the *Allow Custom Password* application setting is set to *True*. The value in the *Password Length* field in application settings is shown for guidance when entering a password. For more information about both of these application settings, see Application Settings: Enrollment Tab on page 609.

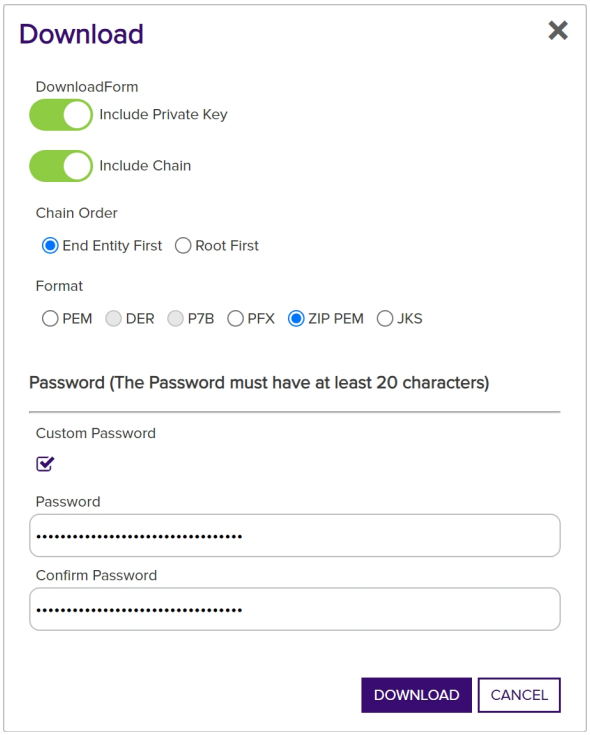


Figure 39: Download a Certificate with Custom Password

8. Click **Download** to begin the download.

To download a certificate that does not have the private key stored in the Keyfactor Command database:

1. Highlight the row in the results grid and right-click.
2. Choose **Download** from the right-click menu.

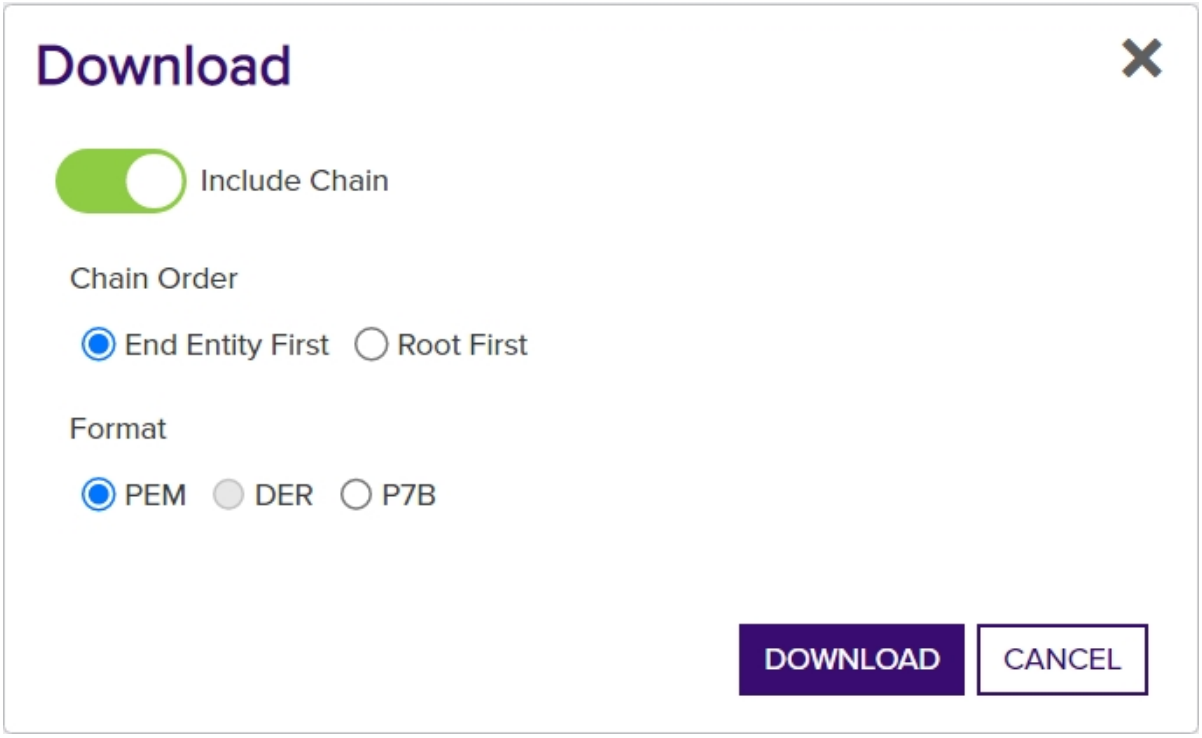


Figure 40: Certificate Operation: Download Certificate without Private Key

3. Chose **Include Chain** to include the certificate chain (root and intermediate certificates) in the download, if required. Include Chain is supported for PEM and P7B outputs.
4. If you selected Include Chain, select a **Chain Order** for the certificates in the resulting output file—either *End Entity First* (at the beginning of the file) or *Root First*. Chain Order is supported for PEM and P7B outputs.
5. Chose an encoding format.

   Note: Selecting the *Include Chain* option changes which formats are available.

6. Click **Download** to begin the download.

Edit (Display)

Select one certificate in the results grid and then click **Edit** at the top of the grid, or **Edit** in the right-click menu, or double-click the row, to pop up the certificate details dialog box in which you can view details of the certificate data and edit metadata fields for the certificate. Users without *Edit Metadata* permissions to certificates will see a **Display** option instead of an **Edit** option.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

Certificates > Collections > Read
Certificates > Collections > Metadata Modify
Certificate Stores > Read

The *Metadata Modify* permission is only needed for users who will be modifying the values of metadata fields for certificates. Users with *Read* permissions may view the exiting metadata values.

The *Read* permission for *Certificate Store Management* is only needed for users who will be viewing values on the Locations tab.

Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. You can use a mixture with, for example, global certificate permissions and container-level certificate store permissions. See Certificate Collection Permissions on page 627 and Container Permissions on page 629 for more information about global vs collection and container permissions.

Note: When you open a certificate for editing, only the custom Keyfactor Command metadata fields are editable.

Note, the certificate details dialog also includes buttons for the **download**, **revoke**, and **renew** (if applicable) operations for users with appropriate permissions. You cannot change any of the certificate attributes from Certificate Authority (shown on the Content tab) or any of the certificate status, validation, locations, or history data tracked by Keyfactor Command (shown on the Status, Validation, Locations and History tabs).

See Certificate Details on page 19 for more detailed information about the certificate details dialog.

If you select multiple certificates to edit at once, only the metadata fields dialog will appear. See **Edit All**.

Edit All

Click **Edit All** at the top of the grid to open the metadata fields for all of the certificates in the query for editing. The button appears active only if no certificates are selected on the grid. All defined metadata fields—including those marked hidden—appear on the Edit All dialog. Each field includes an alert button that identifies whether the certificates in the query have all of same ( ) or different ( ) values for each metadata field. Click the alert button for an explanation of the impact the **Overwrite** settings for this field will have on the certificates.

See Metadata Tab on page 20 for more detailed information about the certificate details metadata.

Click **Allow Modifying** to enable the field for editing. Editing a field and selecting **Overwrite** will change the value for all certificates. Editing this field and not selecting **Overwrite** will only change the value for certificates that do not already have a value defined for this field.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Metadata Modify

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

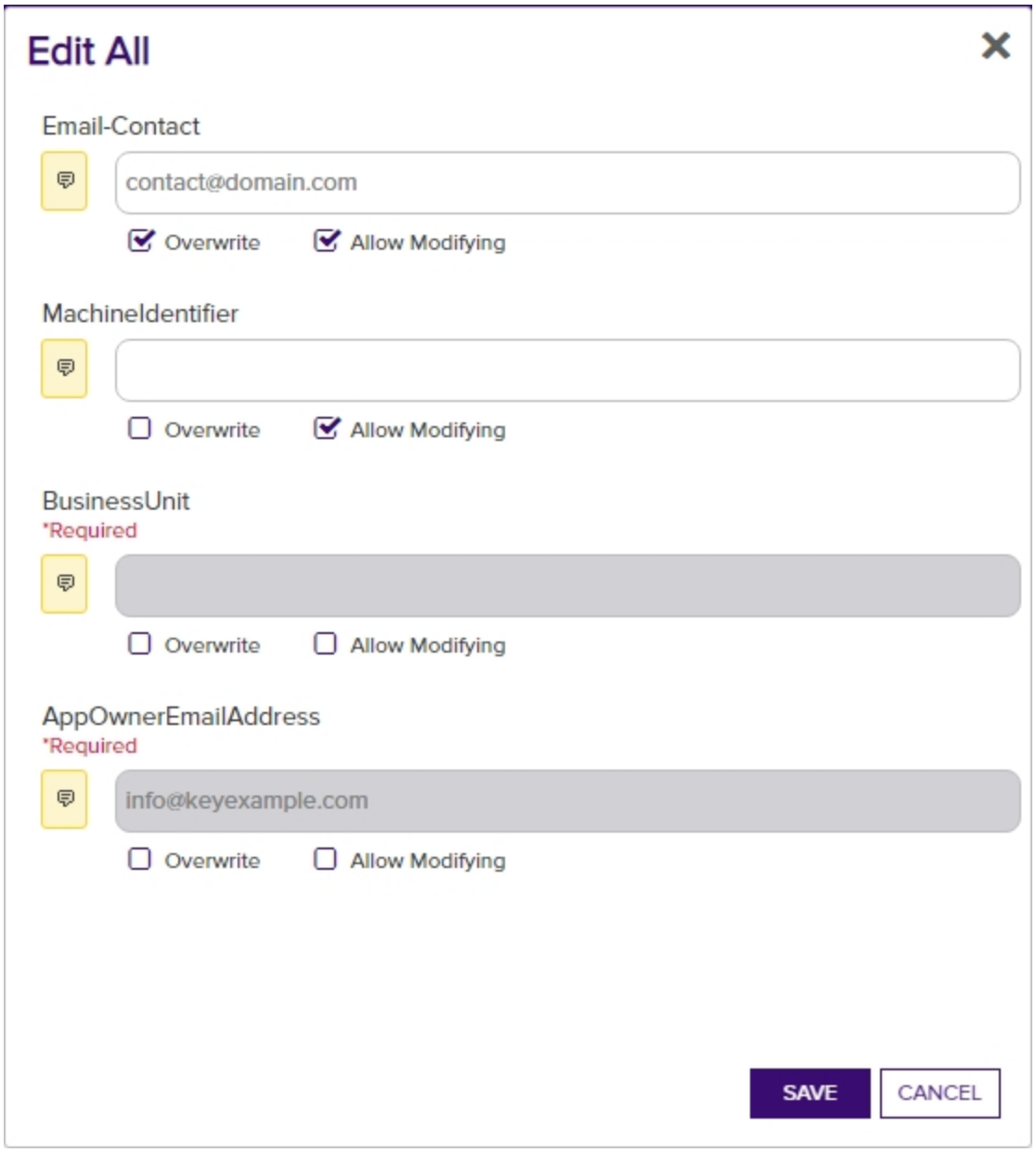


Figure 41: Certificate Operation: Edit All

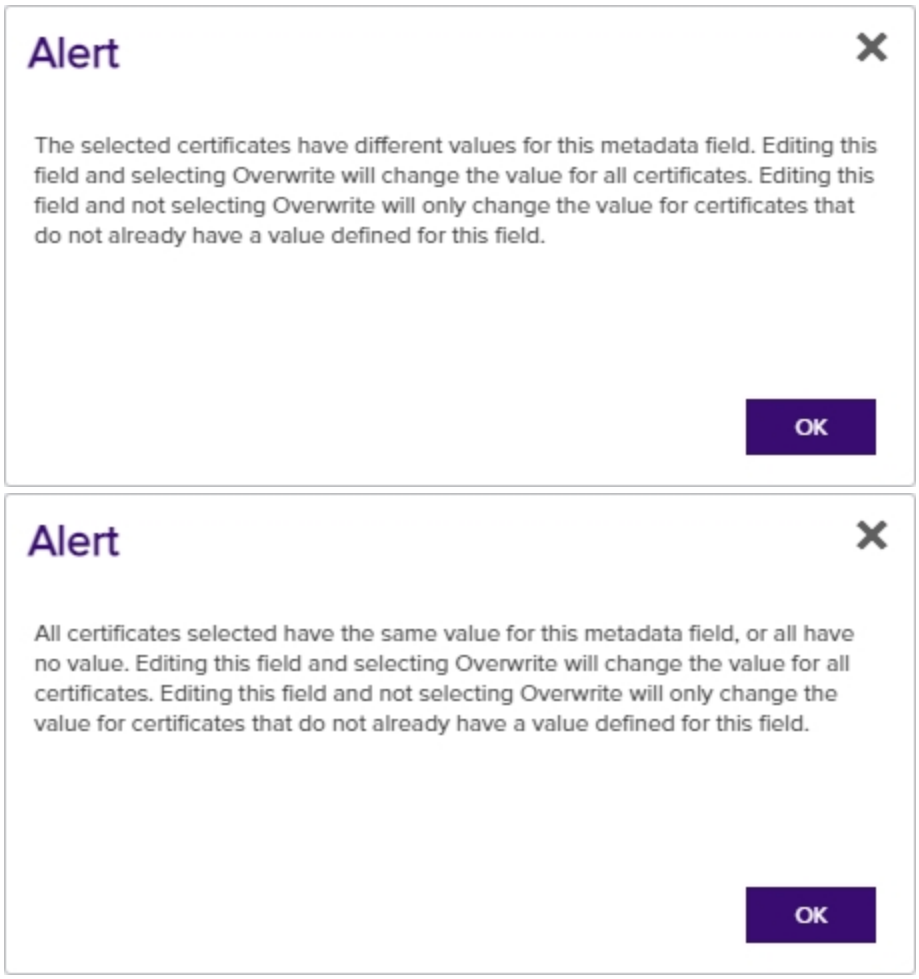


Figure 42: Certificate Operation: Edit All Alerts

Tip: The following setting will need to be configured to run 1+ million certificates in an *Edit All* request. In the IIS Management console, browse to *Default Web Site > Advanced Settings > Limits > Connection timeout*. Set this to a value higher than the default of 120, for example 360.

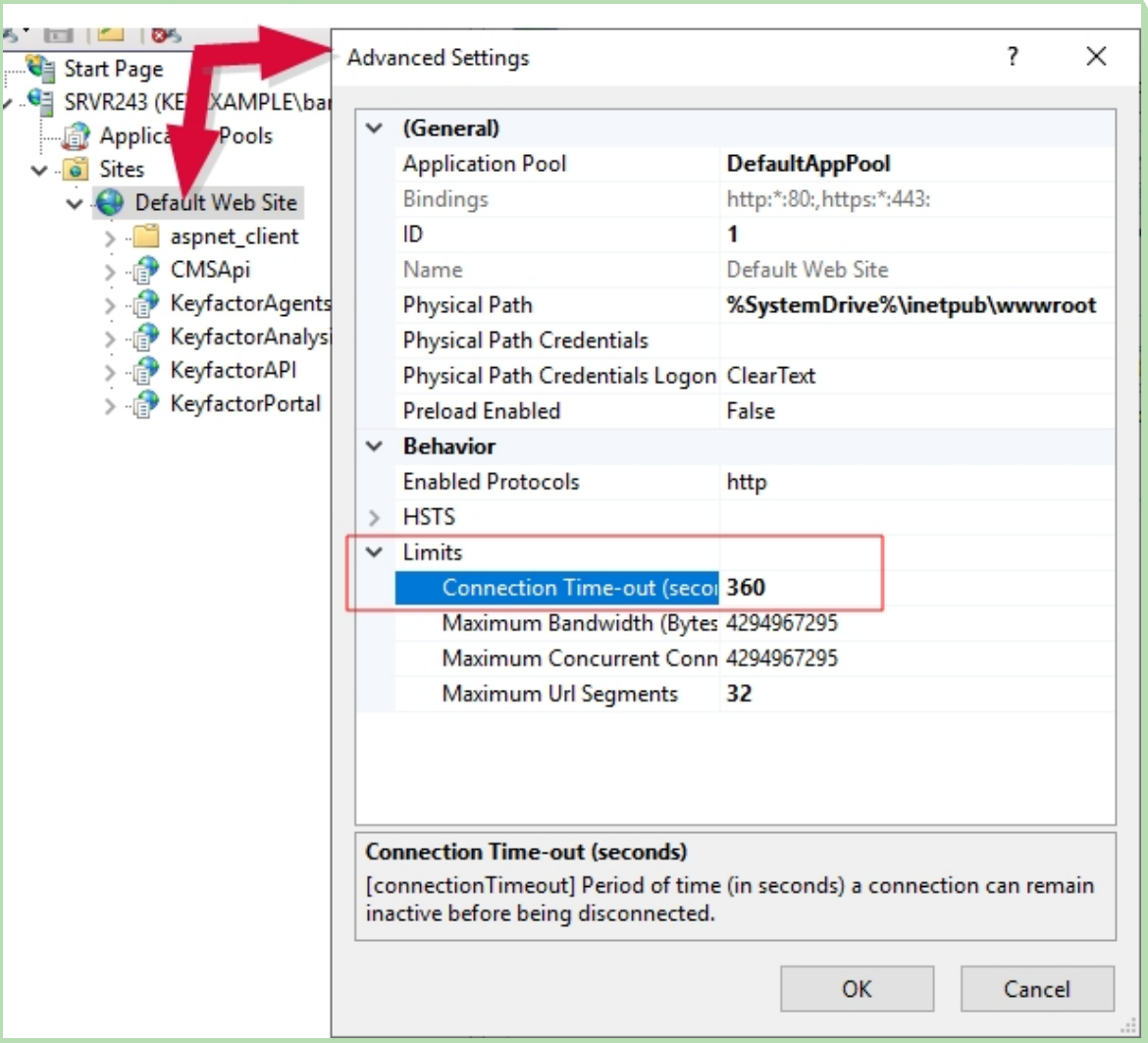


Figure 43: IIS Setting for 1+ Million Records - Certificate Operation: Edit All

Get CSV

Click **Get CSV** from the top of the grid to download all the certificates in the results grid to a comma-delimited CSV file. The button appears active only if no certificates are selected on the grid.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

The CSV file will contain the following information for each exported certificate:

- Issued DN
- Import Date
- Effective Date
- Expiration Date
- Issued CN
- Certificate Authority Name
- Template Display Name
- Principal
- Requester
- Key Type
- Key Size
- Certificate State
- Thumbprint
- Serial Number

A confirmation dialog will pop up providing an approximate file size of the file that will be generated. A CSV file generated from a very large result set may take a long time to download or may be unwieldy to edit.

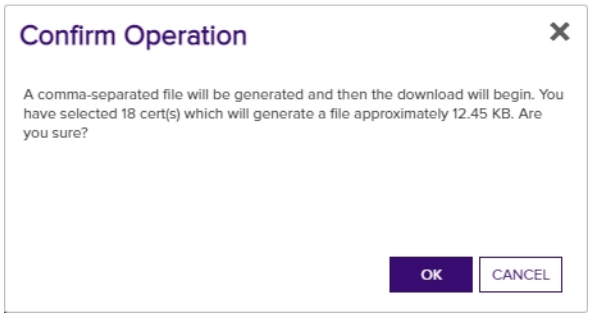


Figure 44: Certificate Operation: CSV Download

Identity Audit

Click **Identity Audit**in the right-click menu to view the certificate level permissions (read, edit metadata, download with private key, revoke, and delete) granted to all user roles defined in Keyfactor Command (see Security Roles and Claims on page 622) for the selected certificate.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Auditing > Read

Certificates > Collections > Read
Security > Read

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

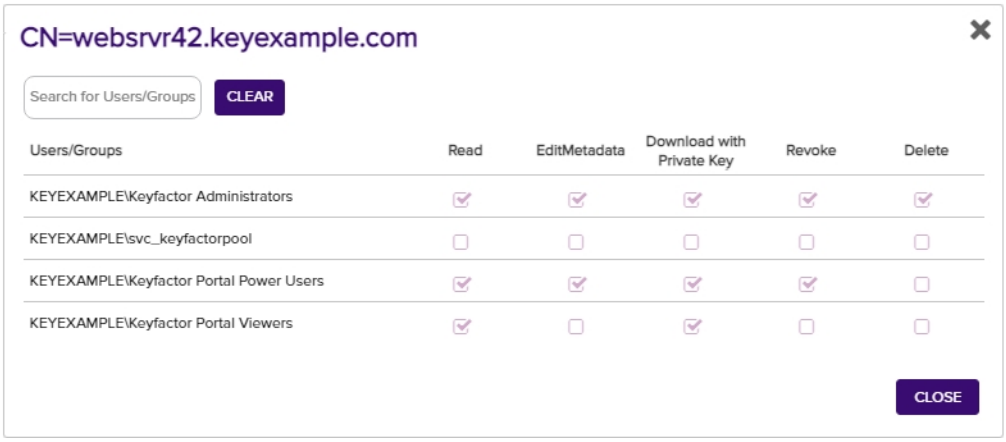


Figure 45: Certificate Operation: Identity Audit

Remove from Certificate Store

Click **Remove from Certificate Store** in the right-click menu to remove the selected certificate from a certificate store or stores. Two dialog boxes will pop up as per Add to Certificate Store on page 45 allowing you to select the certificate store(s) from which you wish to remove the certificate. In the first dialog, select the certificate store from which you want to remove the certificate and click the **Include and Close** button and then click **Save** in the second dialog. Only certificate stores that contain the certificate and to which the user has permissions will be shown.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates Stores > Read
Certificates Stores > Schedule

Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. You can use a mixture with, for example, global certificate permissions and container-level certificate store permissions. See Certificate Collection Permissions on page 627 and Container Permissions on page 629 for more information about global vs collection and container permissions.

Tip: The small *Remove* button at the top of the grid applies to managing the list in the grid only and will remove certificate stores from the selection of stores in the grid. Highlight a row and click *remove* to remove it from the list.

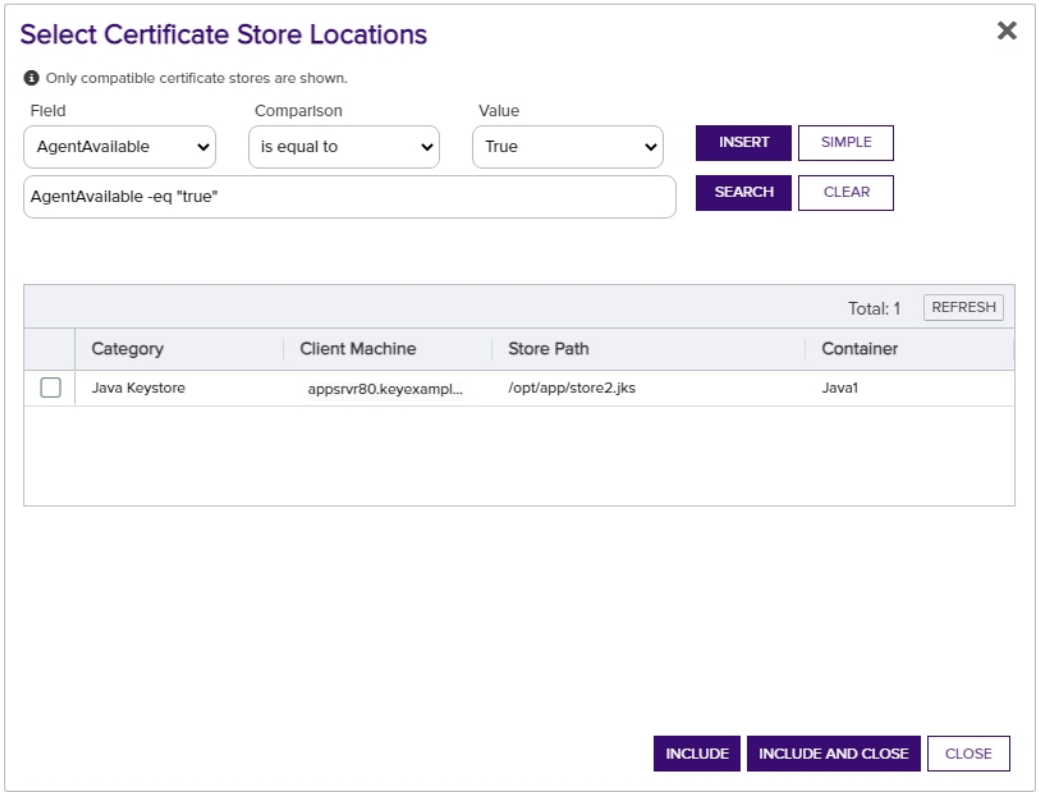


Figure 46: Certificate Operation: Select Stores for Remove from Certificate Store

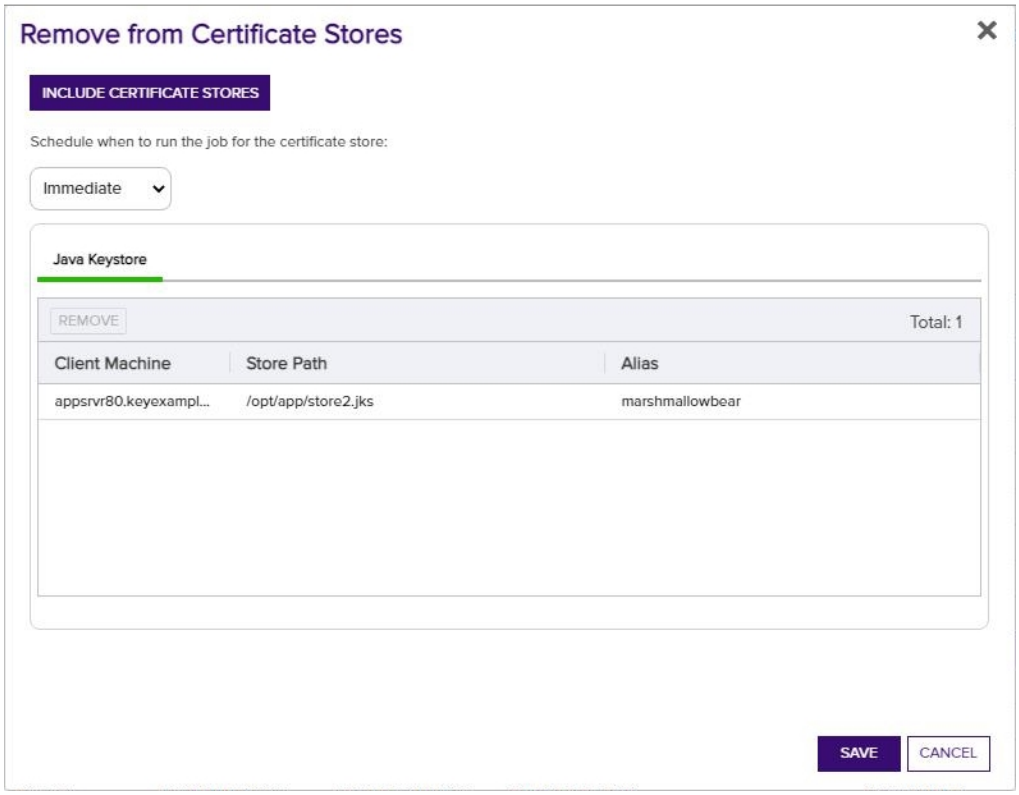


Figure 47:Remove from Cert Store Save Page

Note: When you save this job, a new management job will be added to the orchestrator jobs list. In addition a one-time inventory job will be added immediately following the management job to update Keyfactor Command with the changes to the certificate store. The one-time inventory job does not appear in the orchestrator jobs list in the Management Portal.

Renew

Click **Renew** in the right-click menu to renew or re-issue the selected certificate.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Enrollment > Pfx
Certificates Stores > Read
Certificates Stores > Schedule

Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. You can use a mixture with, for example, global certificate permissions and container-level certificate store permissions. See

Certificate Collection Permissions on page 627 and Container Permissions on page 629 for more information about global vs collection and container permissions.

The renewal dialog includes the options:

- One-click renewal (the **Continue** option), which supports renewal with no further user interaction. If you wish to use *One-Click Renewal* for certificates, the **Allow One-Click Renewals** option must be enabled in both the templates and CAs to which you want *One-Click Renewal* to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354).
- Seeded PFX enrollment (the **Configure** option), to be redirected to the PFX Enrollment page with the information for the certificate pre-populated in the enrollment fields. The **Continue** option is only available if either one of the following is true:
  - The certificate is located *together with its private key* in one or more managed certificate store(s).
  - The certificate was enrolled with a template that has been configured in Keyfactor Command to allow private keys to be encrypted and stored in the Keyfactor Command database (see Certificate Template Operations on page 381).

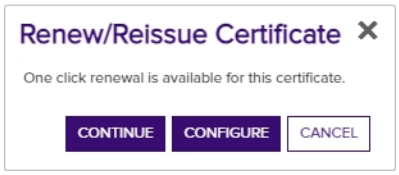


Figure 48: Certificate Operation: Renew/Reissue with the Continue Option

Note: The **Continue** option is only supported if the user performing the renewal has permissions to enroll using the template and CA associated with the original certificate.

From the seeded PFX Enrollment page, you can change the CA or template for enrollment, change the subject information or metadata for the certificate, set or remove SANs, or change the certificate store(s) to which the renewed certificate will be distributed. To change the certificate store(s) for distribution, on the PFX Enrollment page, scroll down to the Certificate Delivery Format section and click the **Include Certificate Stores** button. This will open the *Select Certificate Store Locations* dialog. For more information, see Add to Certificate Store on page 45 and PFX Enrollment on page 146.

Certificates issued by Microsoft CAs will be renewed (meaning the certificate will be issued with a different private key) regardless of how recently they were issued. Certificates issued by other certificate authorities will be renewed (typically retaining the same private key but with a new expiration date) if they are within the renewal window specified by the certificate template and re-issued (retaining the same expiration date) if they are not yet within the renewal window.

Note: Certificate renewal is only supported when attempting to renew the most recently issued certificate in a renewal chain. In other words, if a certificate has been renewed three times resulting in certificates 1, 2 and 3 (1 being the initially issued certificate and 3 being the most recently issued certificate), you can renew certificate 3 but not certificate 2 or 1.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

Note: When you save this job, a new management job will be added to the orchestrator jobs list. In addition a one-time inventory job will be added immediately following the management job to update Keyfactor Command with the changes to the certificate store. The one-time inventory job does not appear in the orchestrator jobs list in the Management Portal.

Revoke

Select one or more certificates in the results grid and then click **Revoke** to revoke the selected certificate(s). When you select revoke, a dialog box pops up prompting for the effective revocation date, the reason for the revocation (for which there are dropdown choices), and comments (required). Upon completion of the revocation, the CRL for the CA in question is immediately republished to reflect the revocation. Unless you choose the revocation reason of *Certificate Hold*, there is no way to undo a revoke so care should be taken with this operation.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Revoke

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Important: In order to successfully revoke certificates, the service account under which the Keyfactor Command application pool is running must be granted “Issue and Manage Certificates” and “Manage CA” permissions to the CA database as per Create Groups to Control Access to Keyfactor Command Features on page 2762 in the *Keyfactor Command Server Installation Guide*, or, if delegation is configured for the CA, the user executing the revoke must have the “Issue and Manage Certificates” permissions while the application pool service account has the “Manage CA” permissions. If you are using explicit credentials to authenticate your CA (see Adding or Modifying a CA Record on page 354), it is the user specified on the CA configuration in Keyfactor Command who must have both these permissions on the CA.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

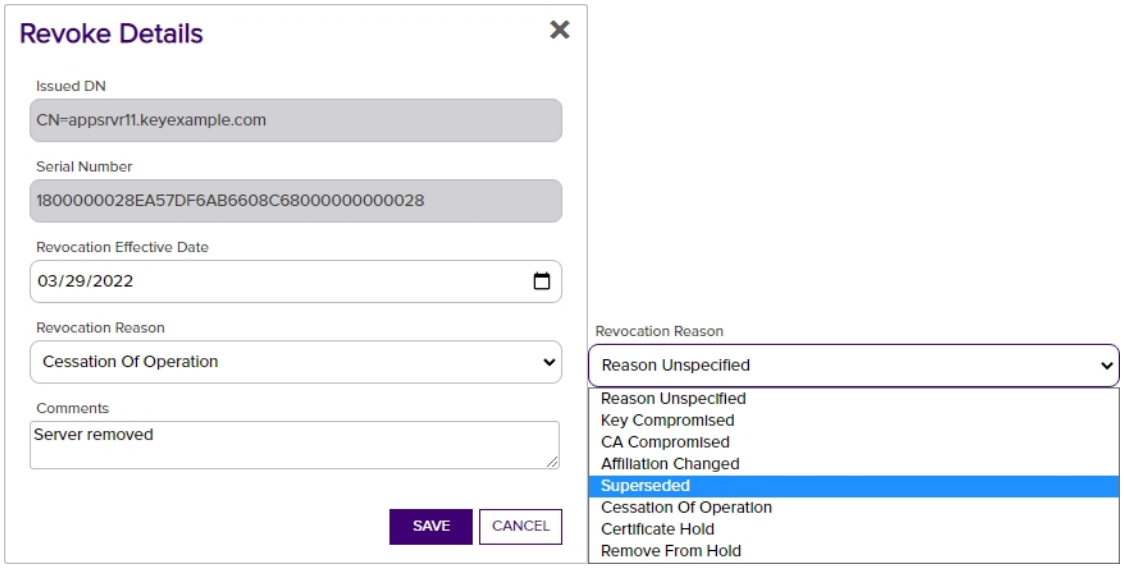


Figure 49: Certificate Operation: Revoke

Revoke: Certificate Hold / Remove from Hold

If you would like to suspend one or more certificates without permanently revoking them, select one or more certificates in the results grid and then click **Revoke** at the top of the grid or **Revoke** on the right-click menu. Select **Certificate Hold** as the revocation reason. You will be required to add a comment in the *Comments* field to **Save** the record change.

When you **Revoke** a certificate using the revocation reason of **Certificate Hold**, the certificate is in the revoked state, with the revocation reason of **Certificate Hold**. You will only be able to see the certificate on a certificate search with *Include Revoked* checked. To return the certificate to the Active state, **Revoke** it again with the reason **Remove from Hold**. You will be required to add a comment in the *Comments* field to **Save** the record change.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Revoke

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

Revoke All

If you would like to revoke ALL the certificates in the current query results set, click **Revoke All** at the top of the grid. The button appears active only if no certificates are selected on the grid.

When you select revoke all, a dialog box pops up prompting for the effective revocation date, the reason for the revocation (for which there are dropdown choices), comments (required), and confirmation of the number of certificates being revoked. In the confirmation field, you must type the suggested message, which includes the number of certificates being revoked, exactly as indicated, including case (e.g. “REVOKE 52” not “revoke 52”).

If any certificates fail revocation, their certificate IDs will be listed in a dialog at the completion of the revocations.

Upon completion of the revocations, the CRL(s) for the CA(s) in question is immediately republished to reflect the revocations. Unless you choose the revocation reason of *Certificate Hold*, there is no way to undo a revoke so care should be taken with this operation.

A maximum of 1000 certificates can be revoked at once with this option. If the query contains more certificates than this, a warning dialog will appear and you will not be allowed to continue with the revocation.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Read
Certificates > Collections > Revoke

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

Note: The Revoke All option can be removed from display on the certificate search pages using the *Revoke All Enabled* application setting (see Application Settings: Console Tab on page 602).

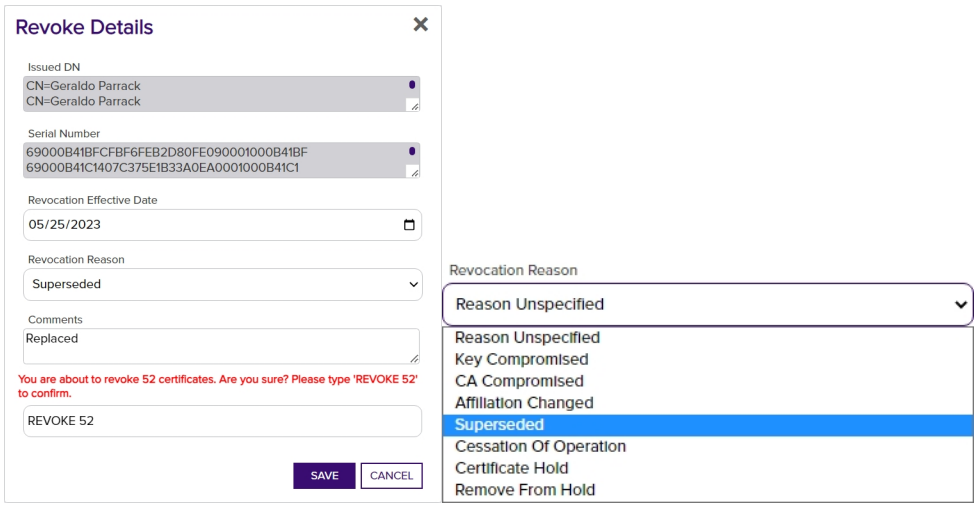


Figure 50: Certificate Operation: Revoke All

2.1.3.4 Add Certificate

The add certificate tool supports importing certificates with the following formats and extensions:

- PEM: .cer or.crt

  If your PEM file has an extension of .pem, rename it to .cer or .crt before using the Add Certificate tool.
- PKCS#12: .pfx or .p12
- PKCS#7: .p7b

This tool has several purposes, including:

- It can be used to import certificates generated outside the enterprise PKI environment—such as those purchased from a commercial certificate vendor or generated by a non-Microsoft or non-EJBCA CA.
- It can be used to import certificates that would not be automatically imported during a synchronization of configured Microsoft or EJBCA CAs such as root CA certificates or certificates with unusual key types (e.g. Dilithium) that aren't supported by synchronization.
- It can be used to import certificates acquired using CSRs generated by Keyfactor Command and issued by a CA not managed using Keyfactor Command to allow for ongoing management with Keyfactor Command.
- It can be used to push a certificate with the associated private key out to a certificate store when you have the appropriate .pfx or .p12 file available.
- It can be used as a quick shortcut to push a certificate without a private key out to a certificate store when you have the certificate file in hand and don't want to search for the certificate in Keyfactor Command in order to push it out to the certificate store.

Before you can add a certificate to a certificate store with this option, you must first add the certificate store in Keyfactor Command (see Certificate Stores on page 408) and install, start, and approve the orchestrator (see Orchestrator Management on page 496 and the Installing Orchestrators on page 2875 guide.

If you import a certificate that has either already been imported via a synchronization task or has been manually imported previously, the certificate will not be re-imported. You will receive a notification message, when you save it, if the certificate already exists in the Keyfactor Command database. Any metadata currently stored in the database for that certificate will be displayed in the metadata fields on the page (for .cer and .crt format certificates), and any changes you make to the metadata on this page will overwrite the existing metadata for the certificate when you complete the import (for all certificate formats).

To use the add certificate tool

1. In the Management Portal, browse to *Certificates > Add Certificate.*
2. In the *Add Certificate* section of the page, click the **Upload** button to open a browse window.
3. In the browse window, browse to select the certificate you wish to import.
4. For a .pfx or .p12 file, when prompted enter the password for the file and **Save**. This will open the Add Certificate page, which will allow you to change/add metadata and choose certificate locations to deploy the certificate to. **Set PFX Password** allows you to reenter the password once you have uploaded the certificate.

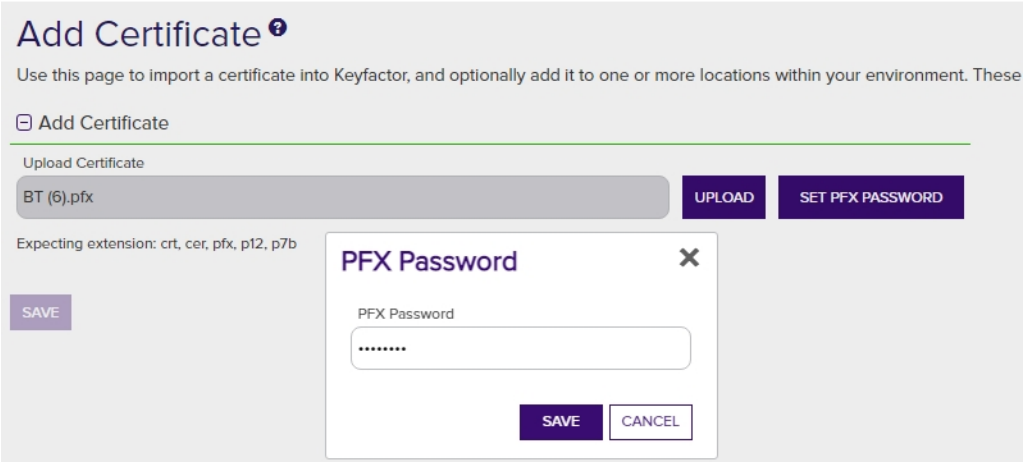


Figure 51: Add Certificate Password for PFX/p12

5. In the *Certificate/PFX Details* section of the page, review the certificate information.

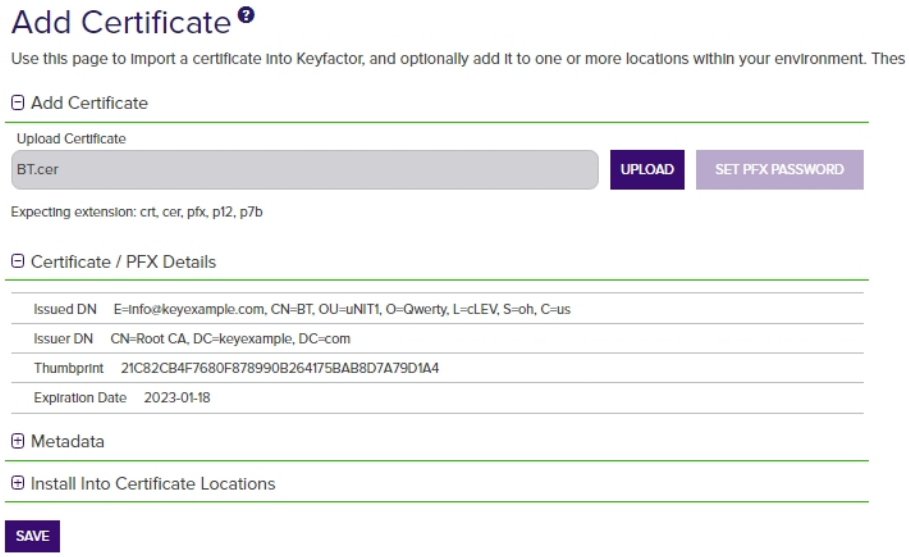


Figure 52: Add Certificate Information

6. In the *Metadata* section of the page, populate the metadata fields as appropriate for the certificate. Metadata fields that have been designated as required on a system-wide or template-level basis will be marked with *Required.

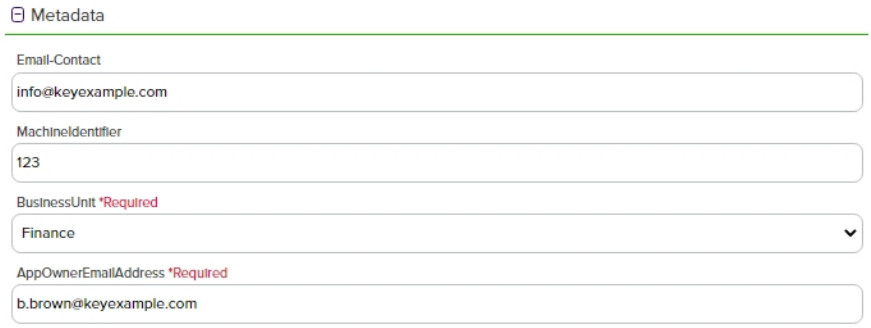


Figure 53: Add Certificate Metadata

7. In the *Install into Certificate Locations* section of the page, select each certificate store location to which you want to distribute the certificate, if desired. To do this, click the **Include Certificate Stores** button. This will cause the *Select Certificate Store Locations* dialog to appear. Make your certificate store selections in this dialog as described in *Select Certificate Store Locations*, below, and click **Include and Close**. You will then see some additional fields on the page. Populate these as per *Add to Certificate Stores* and *Information Required for Certificate Stores*, below.

Select Certificate Store Locations

The *Select Certificate Store Locations* dialog allows you to run queries against your certificate store list to select which store(s) to deploy a selected certificate to. **Check** the box next to each certificate store location to which you want to distribute the certificate.

Note: Only compatible certificate stores and only stores in containers to which you have permissions are shown on the grid.

Tip: You may change the search results by using the search fields at the top of the dialog. All of the Keyfactor Command grid search features are available to assist your search. See Using the Certificate Store Search Feature on page 410 for more information on the available search fields. The default search criteria is *AgentAvailable is equal to True*.

The actions on the *Select Certificate Store Locations* dialog are:

- **Include**

  Click this to add the selected certificate store(s) to your certificate selection and leave the search dialog open for further searches.

- **Include and Close**

  Click this to close the search dialog and add the selected certificate store(s) to your certificate selection, which will then be displayed and ready for updates as per the instructions in *Add to Certificate Stores*.

- **Close**

  Click this to cancel the operation and return to the main page with no certificate stores selected.

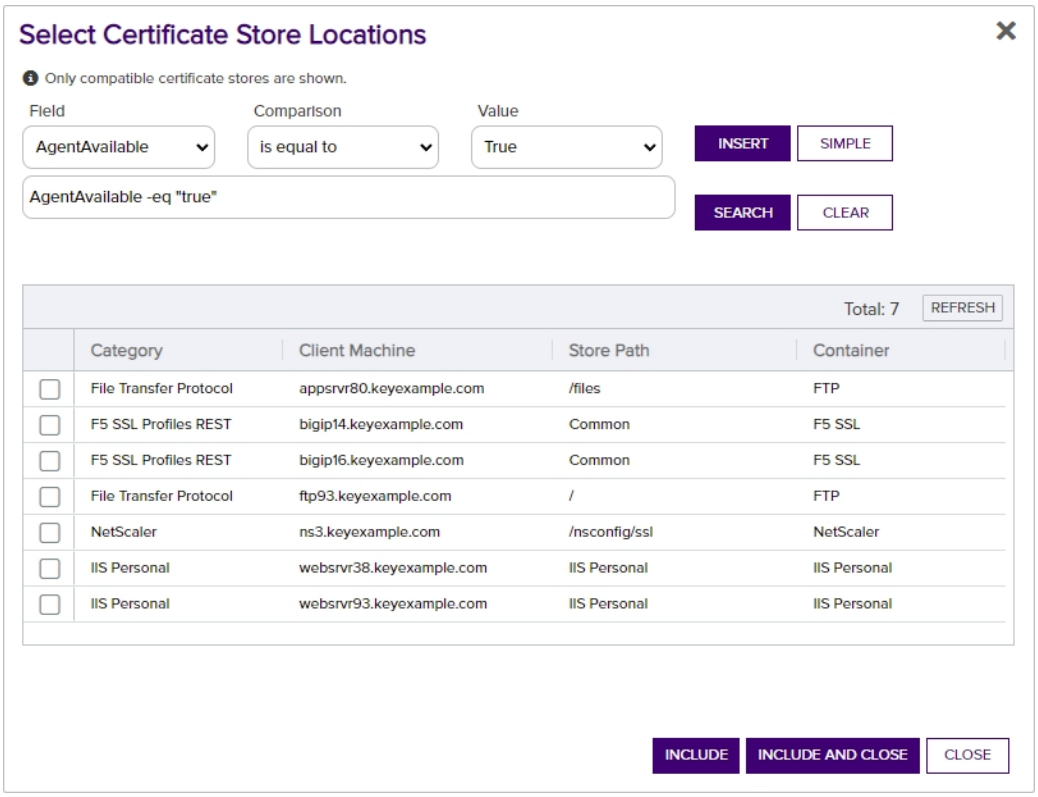


Figure 54: Select Certificate Store Locations Dialog

Add to Certificate Stores

The *Add to Certificate Stores* page appears once you select at least one certificate store to distribute your certificate to. It includes a grid section with a series of tabs that display a tab for each type of certificate store selected with a list of the selected stores under each tab. The header section of the dialog shows global options that apply to the add job as a whole:

- **Include Certificate Stores**

  You may return to the *Select Certificate Store Locations* dialog by clicking **Include Certificate Stores** above the grid. The current selections will be retained.

- **Schedule when to run the job for the certificate store**

  In the **Schedule** dropdown, select a time at which the job to add the certificate to the stores should run. The choices are *Immediate* or *Exactly Once* at a specified date and time. If you choose *Exactly Once*, enter the date and time for the job. A job scheduled for *Immediate* running will run within a few minutes of saving the operation. The default is *Immediate*.

Click **Remove** at the top of the grid to remove the selected certificate store from the page. The certificate will not be added to the store.

For each selected certificate store you can apply the following actions:

- **Overwrite**

  Check **Overwrite** below the grid to overwrite any existing certificate in the same location and with the same name or alias for the selected certificate store type.

- **Alias**

  Add an **Alias** below the grid, if applicable, for the certificate store type. See the **Information Required by Certificate Store** section, below, for more information.

> **Note:** The tab heading of the certificate location will display an alert if an alias is required for the location. If *Supports Custom Alias* is set to **Forbidden** on the certificate store type, the **Alias** field will not display unless *Overwrite* is checked on this page.

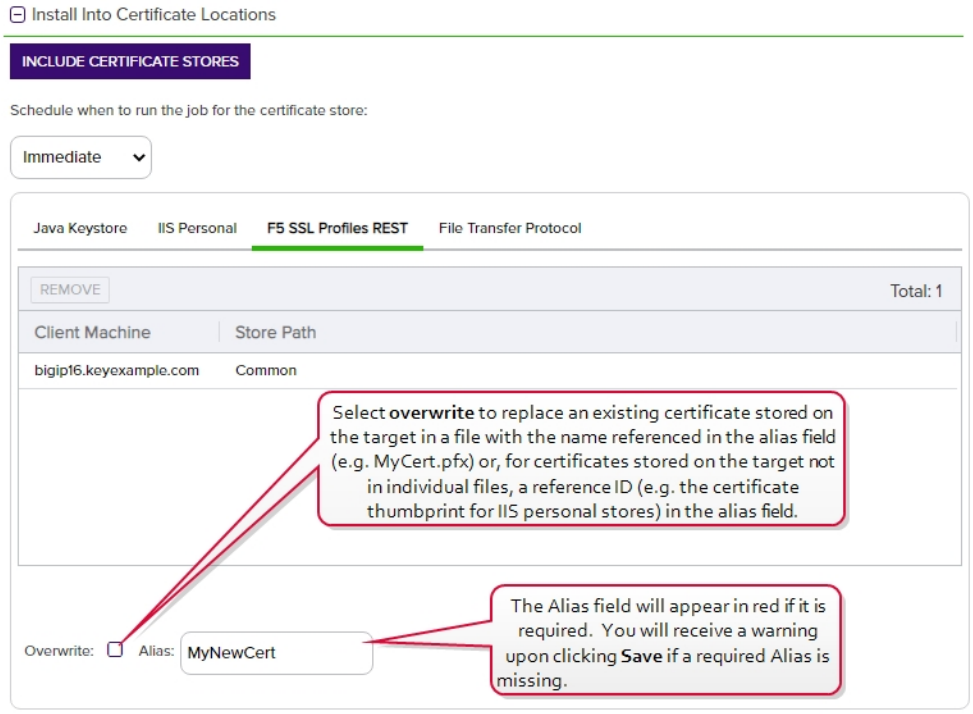


Figure 55: Add Certificate—Install into Certificate Locations

Figure 56: Alias Required Alert on Save

Information Required by Certificate Stores

Each type of certificate store has different requirements for providing an alias or other additional information.

The certificate store type fields that are relevant to certificate store use are:

- **Supports Entry Password**

  If your certificate store type has this enabled, you will have the option to enter a password for the certificate entry in the certificate store on the addition of an entry into the certificate store.

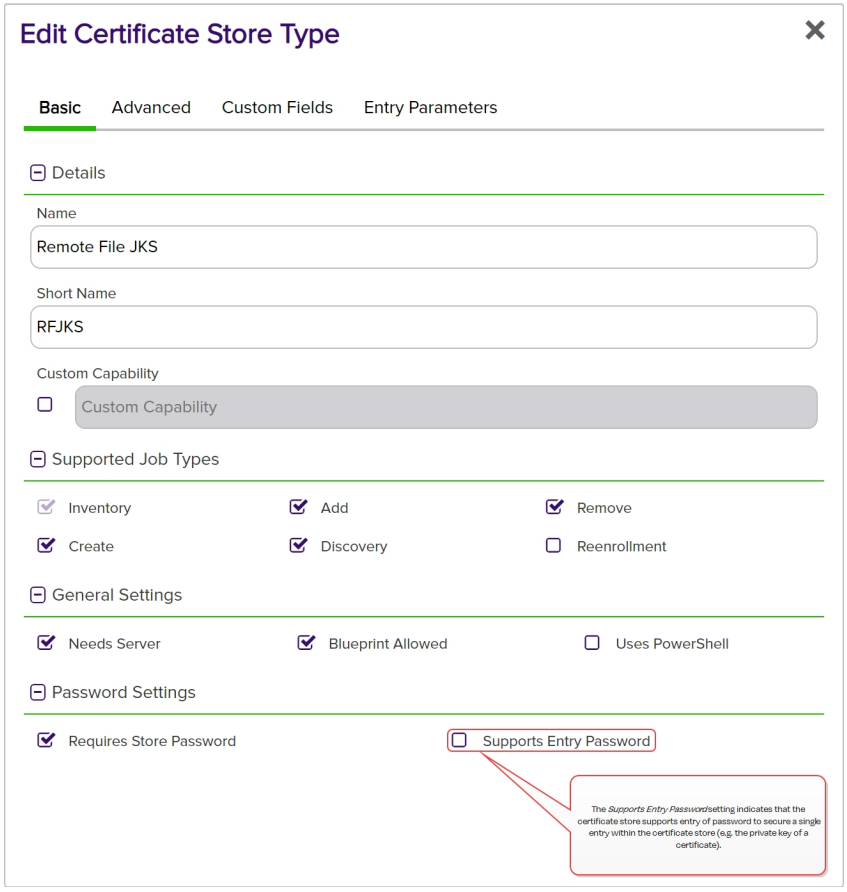


Figure 57: Certificate Store Type Configuration: Basic Tab

- **Supports Custom Alias**

  A value of *Required* indicates that a custom alias will be required when a certificate is added to a certificate store. *Optional* indicates an alias can be associated with the entry if desired. If your certificate store type sets this to *Forbidden*, the *Alias* field will not display when adding a certificate to a certificate store unless *Overwrite* is checked on the add page. In this case, you're not associating an alias with the certificate you're adding to the

store but rather specifying the alias of the certificate already in the store that you wish to replace (in function) with the new certificate you're adding.

The format of custom alias values varies depending on the certificate store type. In many cases, the alias is the thumbprint of the certificate. In some cases, it's the file name of the certificate file or a custom alias provided at the time the certificate was added to the certificate store. For instance:

- For an Amazon Web Services (AWS) store, the alias is the internal ID assigned by Amazon (the Amazon resource number or ARN). Provide the entire contents of the *Alias/IP* from this field when entering an alias for overwrite. For example:

  arn:aws:acm:us-west-2:220531701668:certificate/88e5dcfb-a70b-4636-a8ab-e85e8ad88780

- For F5 stores using the Keyfactor custom-built F5 Certificate Store Manager extension (see Installing Custom-Built Extensions on page 2940), the alias is the file name used to store the file in the device file system, minus the extension (e.g. use alias MyFile for a file named MyFile.crt). Aliases should be entered without spaces. Note that certificate names are case sensitive.

Note: Keyfactor Command will automatically strip out any spaces between the octets in thumbprints in the alias field, so it does not matter whether you enter the thumbprint with or without spaces.

Tip: When adding a certificate to a certificate store, you have the option to overwrite an existing certificate with the current certificate. If you choose this option, you will need to provide the alias of the certificate you wish to overwrite. Find the alias values by navigating to *Management Portal > Certificates > Certificate Search*. Select the certificate you wish to overwrite and double-click, or click **Edit**, from the grid header or right-click menu. Choose the **Locations** tab and double-click on the Location Type (this must have a number other than zero in the *Count* column) to open the details dialog. The *Alias* field holds the information that may be required for an overwrite.

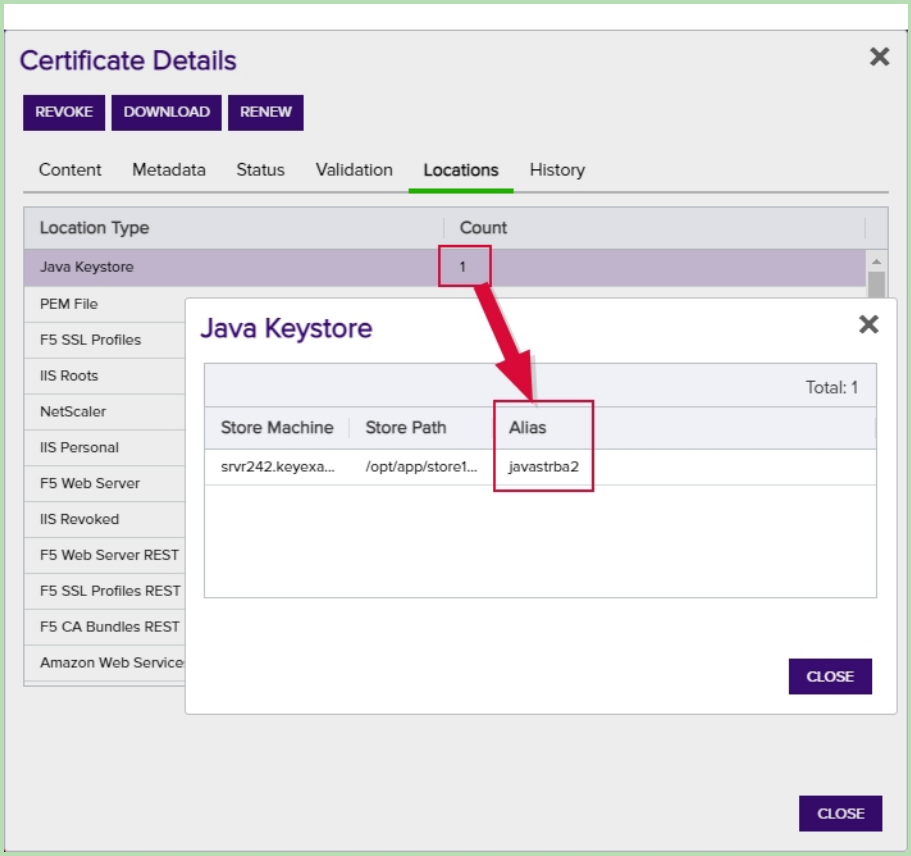


Figure 58: Example: Certificate Location Details for a JKS Location

- **Private Key Handling**

  When adding a certificate to a certificate store, if you select a certificate that does not have an associated private key, certificate stores with this option set to *Required* will not appear as available stores to which the certificate can be added. If this option is set to *Forbidden* and the selected certificate has a private key, the private key will be ignored and only the public key will be delivered to the target.

Note: Private keys are always available in PFX Enrollment.

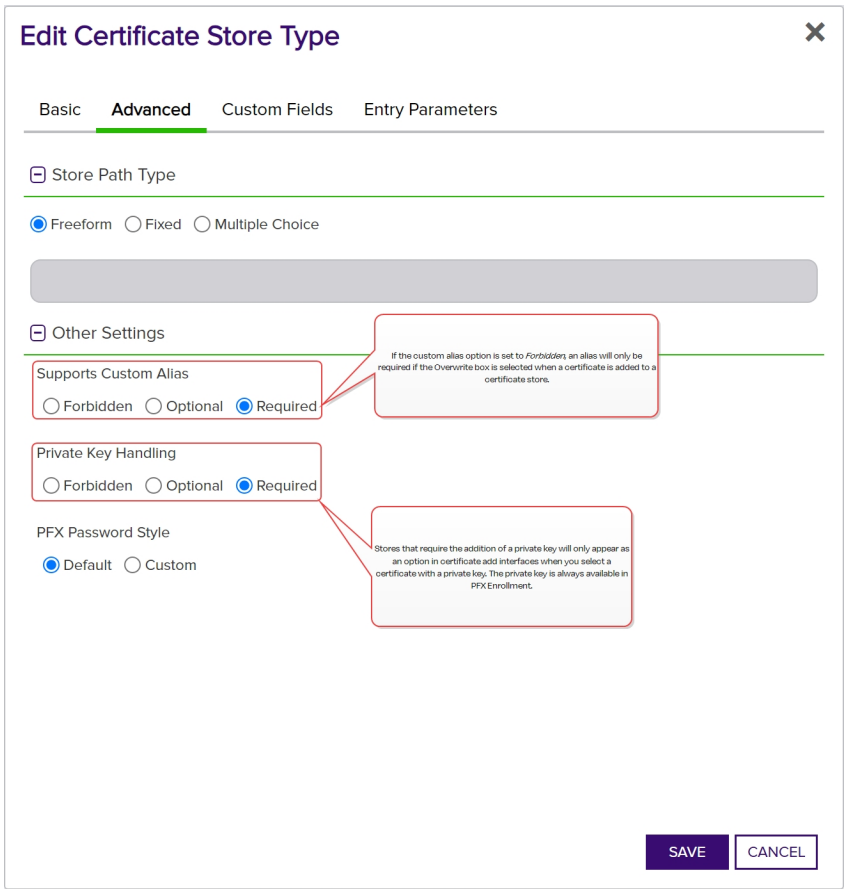


Figure 59: Certificate Store Type Configuration: Advanced Tab

- **Entry Parameters**

  Not all certificate store types will have entry parameters. The ones shown in Figure 60: Certificate Store Type Configuration: Entry Parameters Tab are for the custom *Windows Certificate* type for the Keyfactor custom-built IIS Certificate Store Manager extension (see Installing Custom-Built Extensions on page 2940).

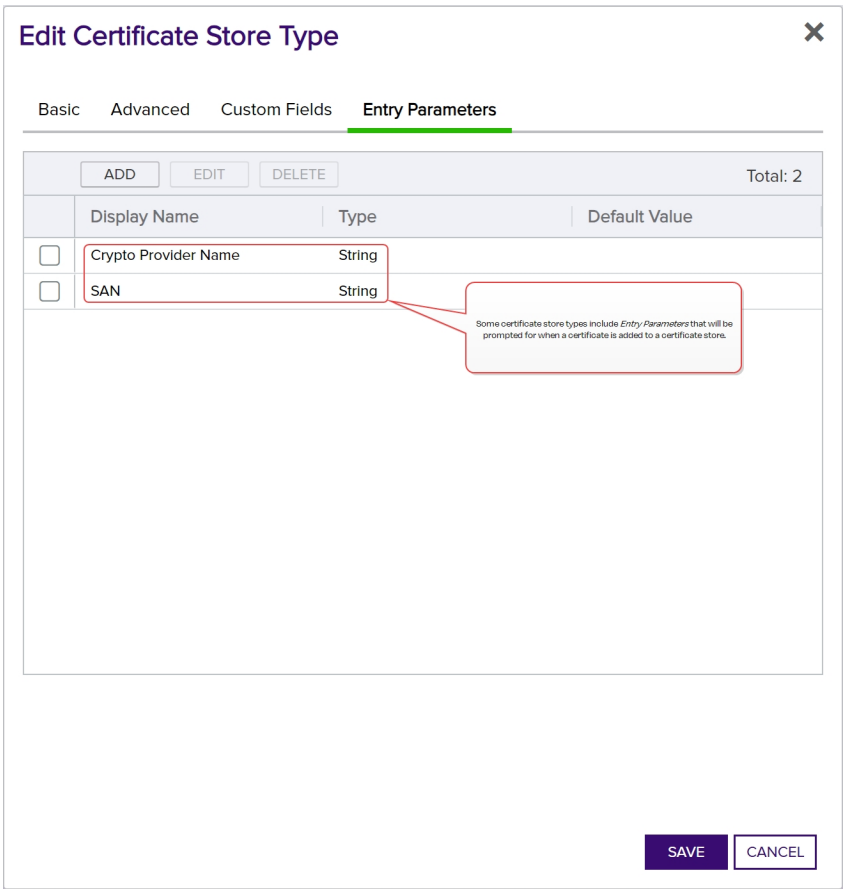


Figure 60: Certificate Store Type Configuration: Entry Parameters Tab

8. Click **Save** to import the certificate to Keyfactor Command

Note: When you import a certificate containing a private key (a .pfx or .p12 file), the private key for that certificate is stored in the Keyfactor Command database. Users with limited permissions to the Add Certificate function may have permissions to upload certificates but not store private keys. If a user with this permission model uploads a certificate containing a private key, the certificate itself will be imported (if it does not already exist in the database), but the private key will not be stored. The user will receive a message indicating this. For more information about setting permissions for importing certificates, see Security Roles and Claims on page 622.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

2.1.3.5 Certificate Collection Manager

The Certificate Collection Manager is used to:

- View a list of certificate collections.
- Change whether or not the collections show in Navigator (the top menu, where they appear under *Certificates*).
- View whether or not the collections show in the dashboard widget (see Dashboard: Collections on page 12).
- Delete a certificate collection.
- Search for specific certificate collections from the list (see Using the Collection Manager Search Feature on page 89).
- View all the certificates in a collection.

  Highlight the collection from the Certificate Collection Manager grid and click the **View** action button. This will open a new window with the name of the collection in a certificate search grid (see Viewing an Existing Certificate Collection on page 87).

Tip: An automated timer service runs against these collections (except *My Certificates*) to monitor certificates entering and leaving the collections so they can be used to trigger a custom workflow to manage your PKI health (see Workflow Definitions on page 230).

To open the Certificate Collection Management grid, browse to *Certificates > Collection Manager* in the Management Portal. The Certificate Collection Management page includes the following collection action buttons from the grid header:

- Set **Show in Navigator** on the collection to determine whether or not the collection appears in Navigator (the top menu under Certificates). To change this setting, highlight the row in the collection management grid and click **Show in Navigator** at the top of the grid, or right-click the collection in the grid and choose **Show in Navigator** from the right-click menu. This will toggle the Yes/No in the **Show in Navigator** grid column.
- To delete a collection, highlight the row (or rows) in the collection management grid and click **Delete** at the top of the grid or right-click the collection in the grid and choose **Delete** from the right-click menu.
- Highlight a row in the collection management grid and click **View** at the top of the grid, or right-click the collection in the grid and choose **View** from the right-click menu to be taken to the list of certificates in that collection. Choosing this option will open the certificate search page in a new window filtered with the specific collection.

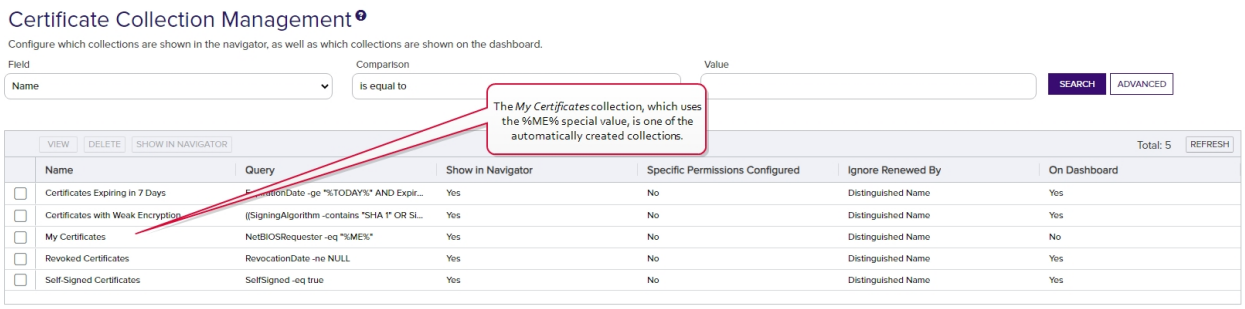


Figure 61: Certificate Collection Manager

Keyfactor Command Auto-Created Collections

Several collections are created automatically when Keyfactor Command is installed.

- Certificates Expiring in 7 Days
  This collection uses the special %TODAY% value in place of the current date to create a collection that can be used on any day to find the certificates that will expire within the next week. Only active certificates are included in this collection. The query for this collection is:

  ```
  ExpirationDate -ge "%TODAY%" AND ExpirationDate -le "%TODAY+7%" AND CertState -eq "1"
  ```

- Certificates with Weak Encryption
  This collection uses a variety of key type, key size, and signing algorithm queries to produce a collection that returns active certificates that have weak encryption. The query for this collection is:

  ```
  ((SigningAlgorithm -contains "SHA 1" OR SigningAlgorithm -contains "SHA1" OR SigningAlgorithm -contains "SHA-1") OR (SigningAlgorithm -contains "MD") OR (KeyType -eq 3 AND KeySize -lt 224) OR (KeyType -eq 1 AND KeySize -lt 2048)) AND CertState -eq "1"
  ```

- My Certificates
  This collection uses the special %ME% value in place of a specific user name to create a collection that any user can use to find the certificates on which they were the requester. The query for this collection is:

  ```
  NetBIOSRequester -eq "%ME%"
  ```

  Note: Certificate collections saved using the %ME% value are *not* supported for use in reports or on the dashboard.

- Revoked Certificates
  This collection returns revoked certificates by querying for certificates that have a non-null revocation date. The *Include Revoked* box is automatically checked for this collection when run. The query for this collection is:

```
RevocationDate -ne NULL
```

- Self-Signed Certificates
  This collection returns all certificates that are self-signed. In environments with no certificates imported from external sources (e.g. SSL scanning), this would typically just be CA certificates. The query for this collection is:

  ```
  SelfSigned -eq true
  ```

Important: The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

Important: All automatically created collections are included on the menu by default, and all are included in the Certificate Collections Management grid by default. They are created for fresh installations of Keyfactor Command only, not upgrades, so as not to overwrite any user-defined collection for existing installations.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Viewing an Existing Certificate Collection

To view an existing certificate collection, either browse to the *Certificates* dropdown on the Management Portal menu and select the desired collection from the dropdown (if the collection has *Show in Navigator* set as **Yes**), or browse to *Certificates > Collection Manager* from the Management Portal and then select **View**, or double-click the row, from the Certificate Collection Management grid. When you select the collection for viewing, the search will begin immediately and the certificate search grid will open with the results from the collection. For information on using the certificate search grid, see Certificate Search Page on page 34.

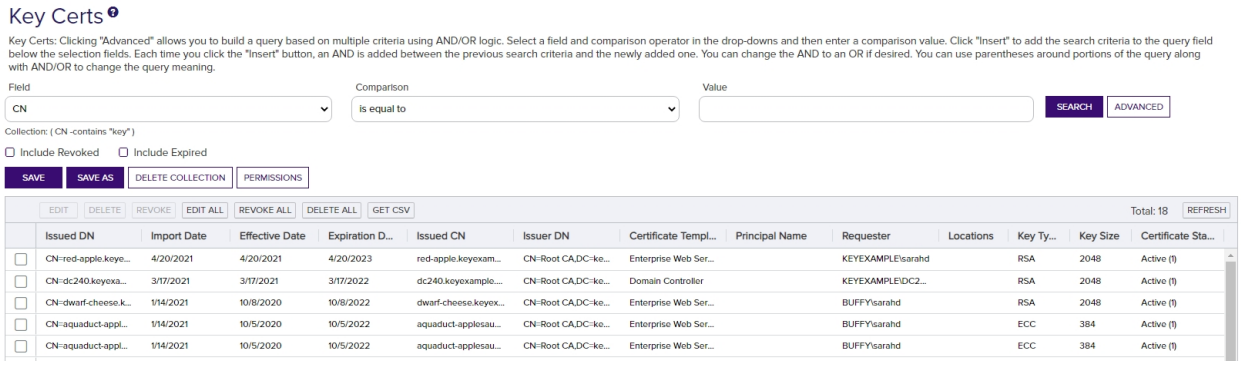


Figure 62: View Collection

When viewing an existing collection, you can further refine the collection query by including additional selection criteria in the query field, but these are used in addition to the base query. You are not allowed to clear the base query for the collection, which is displayed above the advanced query field. For example, for the collection shown in Figure 63: Collection with Query Modification, if the user added this in the query field:

```
CN -notcontains "keyother"
```

The query would return all the certificates issued in the last 30 days with the string appsrvr in the CN using a template referencing web but without the string keyother in the CN—in other words, the web server certificates for application servers issued in the last 30 days for the keyexample.com domain but not the web server certificates for application servers issued in the last 30 days for the keyother.com domain.

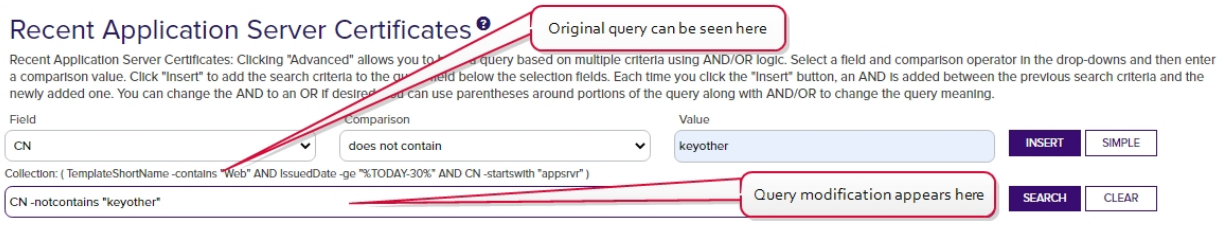


Figure 63: Collection with Query Modification

Available operations on a certificate collection include; **Save**, **Save As**, **Delete Collection** or view **Permissions** on the certificate collection.

Click **Save** to edit the existing collection. You may change the following about the collection from this option:

- The collection *Name*.
- The collection *Description*.
- The collection query *Content*.
- The *Ignore Renewed Cert Results by* setting.
- The *Show on Dashboard* setting.
- The *Show on Navigator* setting.

For more information on these, see Saving Search Criteria as a Collection on page 42.

Note: Certificate collections that are configured for *Certificate Entered Collection* or *Certificate Left Collection* workflows (see Workflow Definition Operations on page 235) cannot be edited. This is done to prevent triggering a large number of entered/left workflows.

Click **Save As** to create a new collection based on the existing collection. You can then edit the search criteria for the new collection without affecting the existing collection. Click **Delete Collection** to delete the certificate collection. Click **Permissions** to view collection level permission for the collection (see Certificate Collection Permissions on page 627).

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

Using the Collection Manager Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

Favorite

Whether the certificate collection has been marked as a favorite collection, meaning it will appear on the Navigator—on the Certificates top-level menu dropdown—true/false.

Name

Complete or partial matches with the name of the certificate collection.

Query

Complete or partial matches with the string that makes up the search criteria for the certificate collection. For example:

```
(IssuedDate -ge \"%TODAY-7%\" AND
TemplateShortName -ne NULL) OR
(IssuedDate -ge \"%TODAY-7%\" AND
IssuerDN -contains \"keyexample\")
```

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

2.1.4 Reports

Keyfactor Command uses the Logi Analytics Platform to provide a number of built-in reports based on certificate data in the Keyfactor Command database. These reports are available for viewing through the Management Portal, if you configured that option during the installation and configuration process (see Dashboard and Reports Tab on page 2807 in the *Keyfactor Command Server Installation Guide*). The reports can also be configured to save to a network path or deliver via email periodically, if desired.

As of Keyfactor Command version 10, Logi has been upgraded to v14 SP2 and a new Logi license is included in the application.

Note: Any CAs that have not been configured for synchronization will not appear as an option for reports which require selecting a CA.

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

Once a report has been generated, you may be able to export it to either PDF, Excel, or CSV. The export file types available for each standard report are shown in Table 3: Chart of Available Exports per Standard Report.

Table 3: Chart of Available Exports per Standard Report

PDF and Excel	Excel and CSV	PDF, Excel and CSV
Certificate Count by Template	Certificates Found at TLS/SSL Endpoints	Certificate Count Grouped by Single Metadata Fields
Certificate Count by User per Template	Certificates in Collection	
Certificate by Key Strength	Expiration Report by Days	
Certificates by Revoker	Full Certificate Extract	
Certificates by Type and Java Keystores	Revoked Certificates in Certificate Stores	
Certificate Issuance Trends with Metadata	SSH Keys with Root Logon Access	
Expiration Report	SSH Trusted Public Keys with No Known Private Key	
Issued Certificates Per Certificate Authority	SSH Key Usage Report	
Monthly Executive Report		
PKI Status for Collection		
Statistical Report		
SSH Keys by Age		

Report Drill-down

Most reports now have drill-down capability. Clicking on a chart or graph segment in a report will open the corresponding query grid in a new browser window or tab populated with the query as defined by the selected graph segment. For example, for the *Certificates by Key Strength* report, clicking on a bar or pie will take you to the Certificate Search page pre-populated with the query that corresponds to that bar or pie.

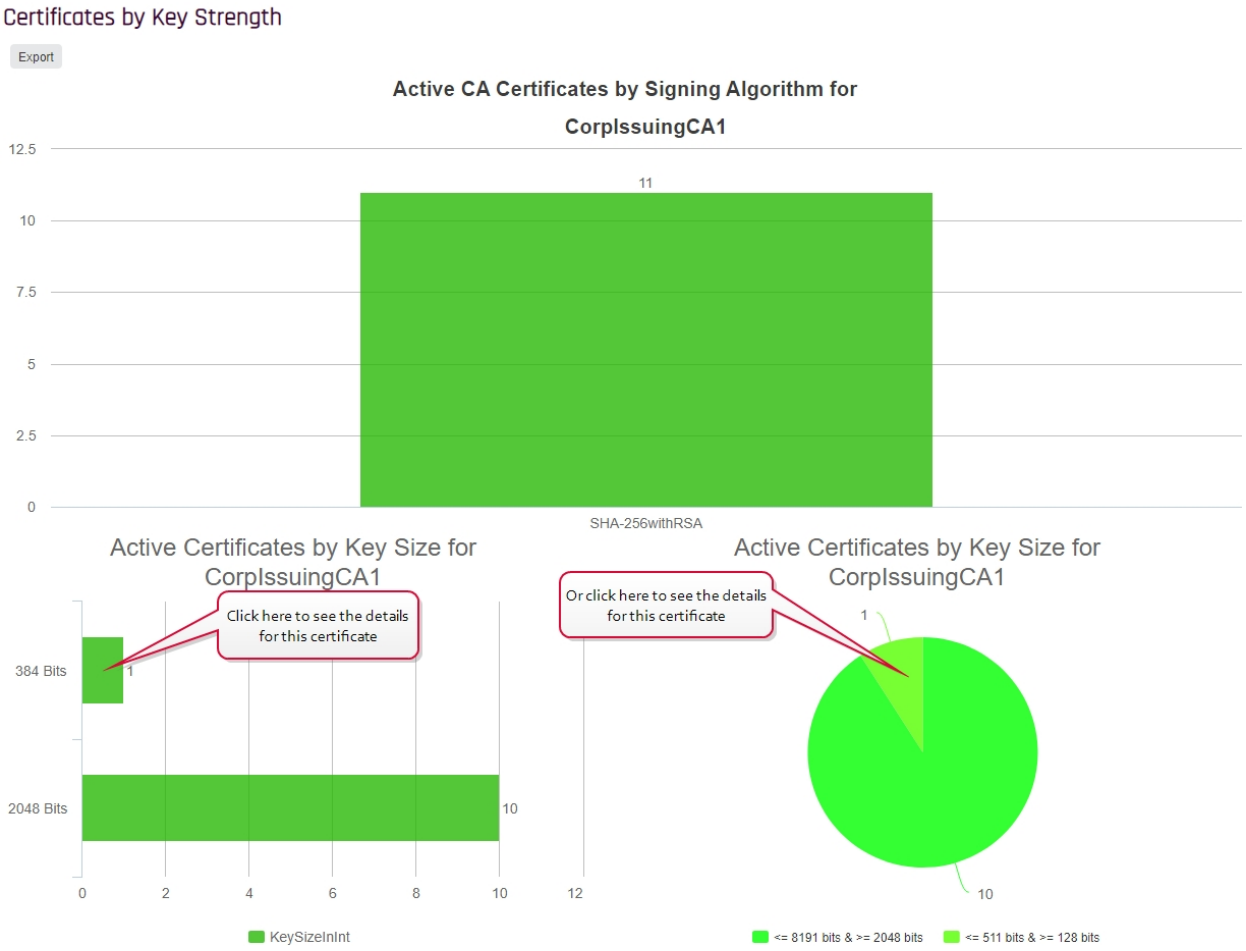


Figure 64: Report Drill Down: Certificates by Key Strength Report

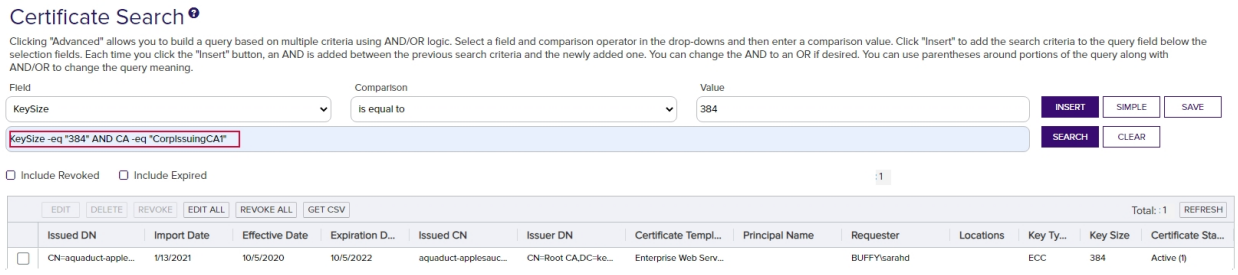


Figure 65: Report Drill Down: Certificate Search Results

List of Built-In Reports

The following reports are available as part of the standard Keyfactor Command installation. Those marked with a (*) have been configured to *Show in Navigator* by default, so they appear on the Management Portal top menu under Reports. The Report Manager page shows all the available reports.

- Certificate Count by Template
- Certificate Count by User per Template
- Certificate Count Grouped by Single Metadata Field
- Certificate Issuance Trends with Metadata
- Certificates by Key Strength
- Certificates by Revoker
- Certificates by Type and Java Keystores
- Certificates Found at TLS/SSL Endpoints
- Certificates in Collection (*)
- Expiration Report (*)
- Expiration Report by days (*)
- Full Certificate Extract (*)
- Issued Certificates per Certificate Authority
- Monthly Executive Report
- PKI Status for Collection (*)
- Revoked Certificates in Certificate Stores
- SSH Key Usage Report
- SSH Keys by Age
- SSH Keys with Root Logon Access
- SSH Trusted Public Keys with No Known Private Keys
- Statistical Report (*)

2.1.4.1 Certificate Count by Template

The Certificate Count by Template report includes bar graphs showing the number of certificates issued, failed and revoked by template in the selected date range for the selected CA(s). Separate graphs are generated for issued and revoked certificates and for each selected CA. Each graph contains all the templates that have had certificates issued or revoked for the period.

Note: Clicking on a bar on the graph, or a section of a pie chart or line graph, will open a new window as a drill down to the certificate search grid filtered for the exact criteria of that aspect of the graph. You can return to the original report by navigating to the original report window.

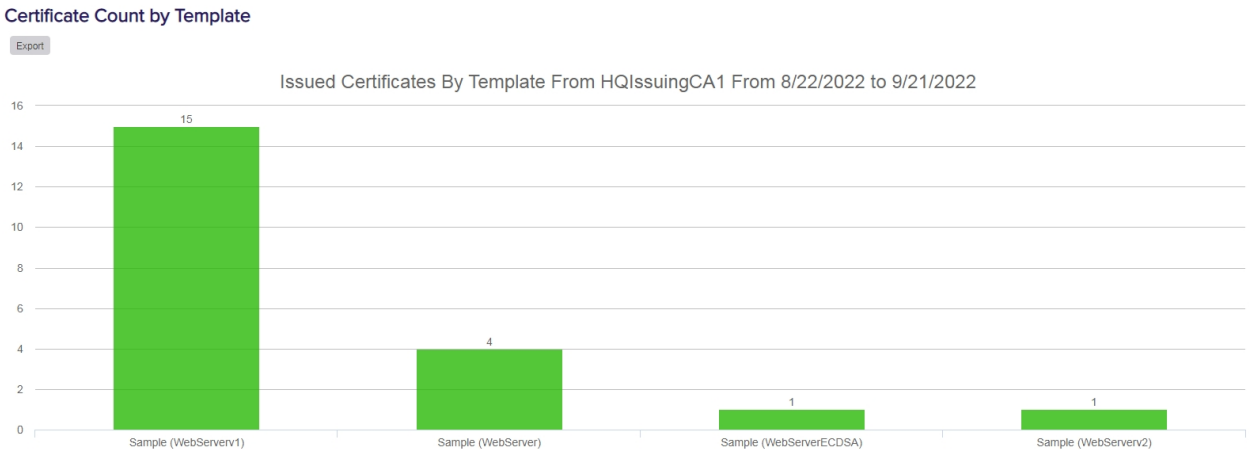


Figure 66: Certificate Count by Template: Issued Certificates

The export options for the Certificate Count by Template report are Excel and PDF.

The input parameters for this report are:

- The start date and end date for the report date range. The default date range is 30 days prior through the current date, meaning only certificates issued and revoked in that date range will be included in the report.
- The CA(s) to include in the report. Templates that are available for issuance from more than one CA are reported separately by CA.

Note: Only CAs configured for synchronization are available for reporting.

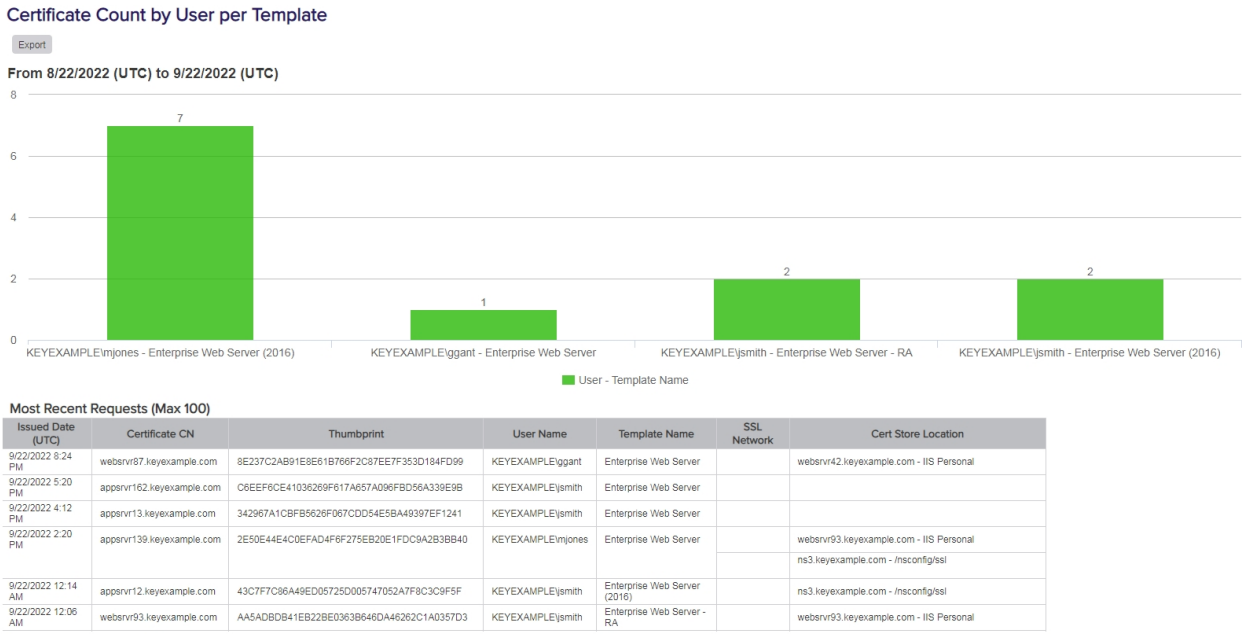
Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

2.1.4.2 Certificate Count by User per Template

The Certificate Count by User per Template report includes a table and bar graphs.

The bar graphs show the number of certificates issued by the certificate requester and template in the selected date range for the selected template(s). The report shows one bar for each requester and template combination; for example, *KEYEXAMPLE\jsmith - Template One* would be one bar and *KEYEXAMPLE\mjones - Template One* would be another bar.

Note: Clicking on a bar on the graph, or a section of a pie chart or line graph, will open a new window as a drill down to the certificate search grid filtered for the exact criteria of that aspect of the graph. You can return to the original report by navigating to the original report window.



Most Recent Requests (Max 100)

Issued Date (UTC)	Certificate CN	Thumbprint	User Name	Template Name	SSL Network	Cert Store Location
9/22/2022 8:24 PM	websrvr87.keyexample.com	8E237C2AB91E8E61B766F2C87EE7F353D184FD99	KEYEXAMPLE\ggant	Enterprise Web Server		websrvr42.keyexample.com - IIS Personal
9/22/2022 5:20 PM	appsrvr162.keyexample.com	C6EEF6CE41036269F617A657A096FBD56A339E9B	KEYEXAMPLE\jsmith	Enterprise Web Server		
9/22/2022 4:12 PM	appsrvr13.keyexample.com	342967A1CBFB5626F067CDD54E5BA49397EF1241	KEYEXAMPLE\jsmith	Enterprise Web Server		
9/22/2022 2:20 PM	appsrvr139.keyexample.com	2E50E44E4C0EFAD4F6F275EB20E1FDC9A2B3BB40	KEYEXAMPLE\mjones	Enterprise Web Server		websrvr93.keyexample.com - IIS Personal
						ns3.keyexample.com - /nsconfig/ssl
9/22/2022 12:14 AM	appsrvr12.keyexample.com	43C7F7C86A49ED05725D005747052A7F8C3C9F5F	KEYEXAMPLE\jsmith	Enterprise Web Server (2016)		ns3.keyexample.com - /nsconfig/ssl
9/22/2022 12:06 AM	websrvr93.keyexample.com	AA5ADBDB41EB22BE0363B646DA46262C1A0357D3	KEYEXAMPLE\jsmith	Enterprise Web Server - RA		websrvr93.keyexample.com - IIS Personal

Figure 67: Certificate Count by User by Template

The table shows detailed information for the certificates issued in the selected time-frame (up to a maximum of 100).

The export options for the Certificate Count by User per Template report are Excel and PDF. The PDF exports in landscape format to accommodate the width of the report.

The certificate details grid includes these fields:

- Issued Date
  The certificate's effective date.
- Certificate CN
  Common name of the certificate.
- Thumbprint
  Thumbprint of the certificate.
- User Name
  The user who requested the certificate. In some cases (e.g. enrollment using the *Restrict Allowed Requesters* option), this will be a service account rather than an end user.
- Template Name
  Name of the template used for the certificate.
- SSL Network
  The name of the SSL network containing the endpoint at which the certificate is found, if any.
- Cert Store Location
  The certificate store or stores in which the certificate is found, if any.

The input parameters for this report are:

- The template names on which to report. Although you can select multiple templates, selecting more than one or two templates can make for a messy report, depending on how many unique users have requested certificates using the selected template(s) in the date range. Defaults for the template(s) on which to report can be configured in the report parameters (see Report Manager Operations on page 127).

- The start date and end date for the date range on which to report. The default date range is one month ending with the current date. These defaults can be changed in the report parameters (see Report Manager Operations on page 127).
- A requester and template combination bar will only be included in the bar chart, with corresponding details in the details grid, if the number of certificates issued for it in the selected date range exceeds the value selected for *Certificate count more than*.

  Example: You want to track down instances of duplicate certificates where user X has been issued a certain type of certificate more than once and more than one of these certificates is still valid (not revoked). To use this report for that, select the template or templates used for that particular type of certificate (say, a client authentication template), select a date range that would cover the full lifetime for certificates issued by that template, and select a value of 1 or greater in the *Certificate count more than* field. The report results will include all users who have multiple certificates issued with the selected template(s) in the selected date range.

Note: Certificates must have a certificate state of *Active* to be included in the report. The report output includes active and expired certificates but not revoked certificates.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

2.1.4.3 Certificate Count Grouped by Single Metadata Field

The Certificate Count Grouped by Single Metadata Field report includes a data table with two columns:

- Metadata Value
  All the populated values for the selected metadata field for certificates issued in the selected date range.
- Certificate Count
  The number of certificates issued for each metadata value in the selected date range.

For example, if the selected metadata field is AppOwnerEmailAddress, the table will show a row for each unique email address populated in a certificate issued in the selected date range with a count of how many certificates share that same email address.

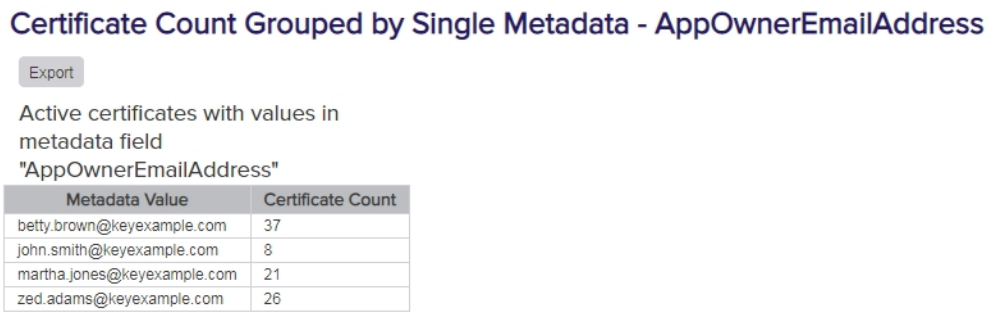


Figure 68: Certificate Count Grouped by Single Metadata Field

The export options for the Certificate Count Grouped by Single Metadata Field report are CSV, Excel, and PDF.

The input parameters for this report are:

- The metadata field on which to report. Only Boolean, integer, multiple choice, and string fields are available for reporting.
- The start date and end date for the date range on which to report. The default date range is one month ending with the current date. Only certificates issued within the time span will be counted.

Note: Certificates must have a certificate state of *Active* to be included in the report. The report output includes active and expired certificates but not revoked certificates. Only certificates issued within the time span will be counted.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

2.1.4.4 Certificate Issuance Trends with Metadata

The Certificate Issuance Trends with Metadata report produces tables and pie charts showing currently active certificates based on the selected input parameters as follows: the number of certificates per requester and the number of certificates per metadata value for each of the metadata fields chosen, based on the certificate collection chosen. Multiple tables and charts will be produced when the report is generated.

Note: Clicking on a bar on the graph, or a section of a pie chart or line graph, will open a new window as a drill down to the certificate search grid filtered for the exact criteria of that aspect of the graph. You can return to the original report by navigating to the original report window.

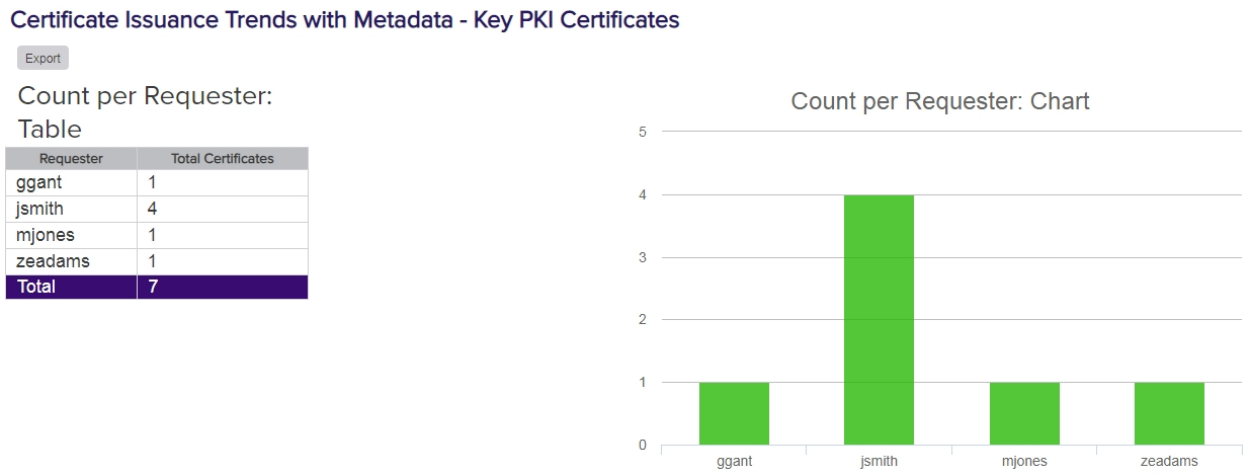


Figure 69: Certificate Issuance Trends with Metadata: Requesters

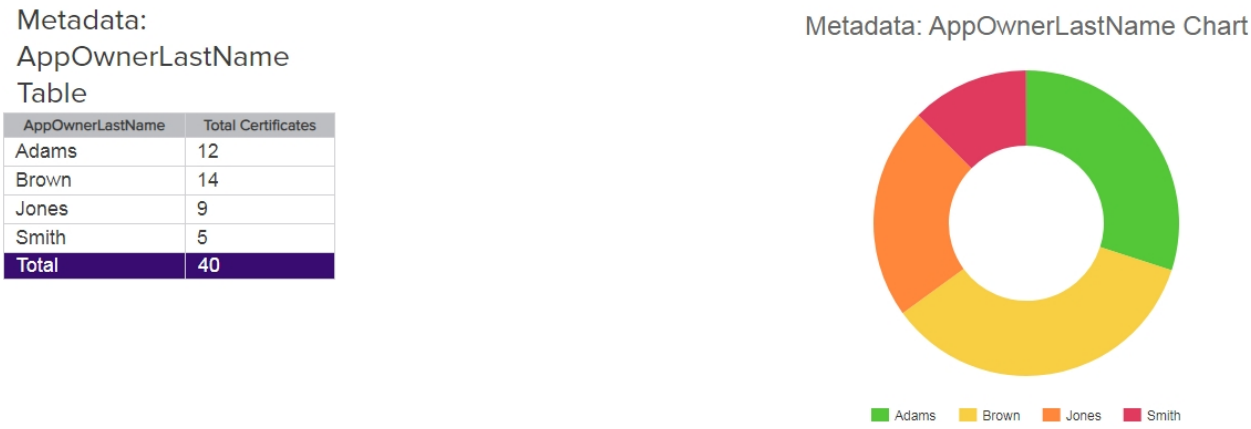


Figure 70: Certificate Issuance Trends with Metadata: Metadata Table and Chart

The export options for the Certificate Issuance Trends with Metadata report are Excel and PDF.

Note: When either scheduling or exporting this report as an Excel file, the output will not include the graphs.

The input parameters for this report are:

- Collections: The name of the collection to report on.
- The start date and end date for the report: The definition of the date range for the report.
- Metadata: Check a metadata field from the pop-up to select it for this report.
- Requesters: A comma-separated list of requester user names (do not included the domain name).

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display

certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

Note: Other than the option of certificates with no associated CA, only CAs currently configured for synchronization are available for reporting.

2.1.4.5 Certificates by Key Strength

The Certificates by Key Strength report includes a bar graph showing the number of active certificates by key strength (e.g. sha-1, sha-256) for the selected CA(s), a bar graph showing the number of active certificates by key size for the selected CA(s), and a pie chart for each selected CA showing the active certificates by key size (e.g. 1024 bit, 2048 bit).

Note: Clicking on a bar on the graph, or a section of a pie chart or line graph, will open a new window as a drill down to the certificate search grid filtered for the exact criteria of that aspect of the graph. You can return to the original report by navigating to the original report window.

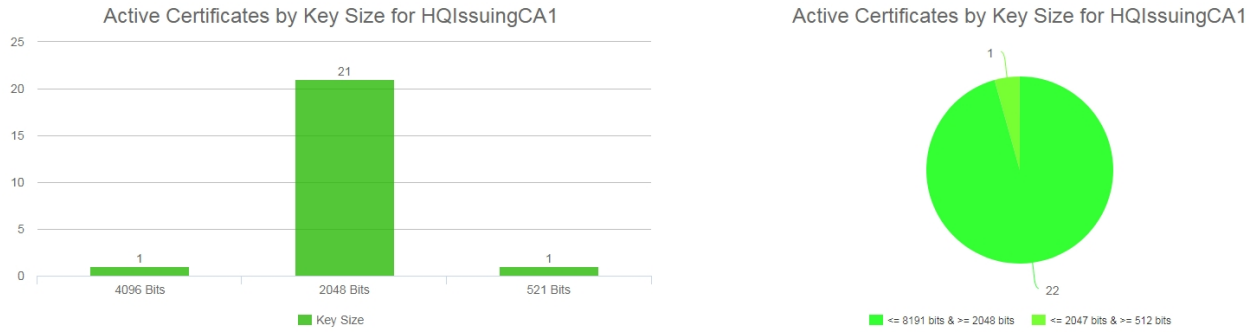


Figure 71: Certificates by Key Strength

The export options for the Certificates by Key Strength report are Excel and PDF.

This report takes as an input parameter the CA(s) on which to report and includes the option to report on certificates that have no associated CA. Typically, these would be certificates found via SSL scanning or inventory on certificate stores.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

Note: Other than the option of certificates with no associated CA, only CAs currently or previously configured for synchronization are available for reporting.

2.1.4.6 Certificates by Revoker

The Certificates by Revoker report includes a bar graph showing the number of certificates revoked through Keyfactor Command in the selected date range for the selected CA(s) broken down by the user doing the revocation. The report shows one bar for each revoker.

Note: Clicking on a bar on the graph, or a section of a pie chart or line graph, will open a new window as a drill down to the certificate search grid filtered for the exact criteria of that aspect of the graph. You can return to the original report by navigating to the original report window.

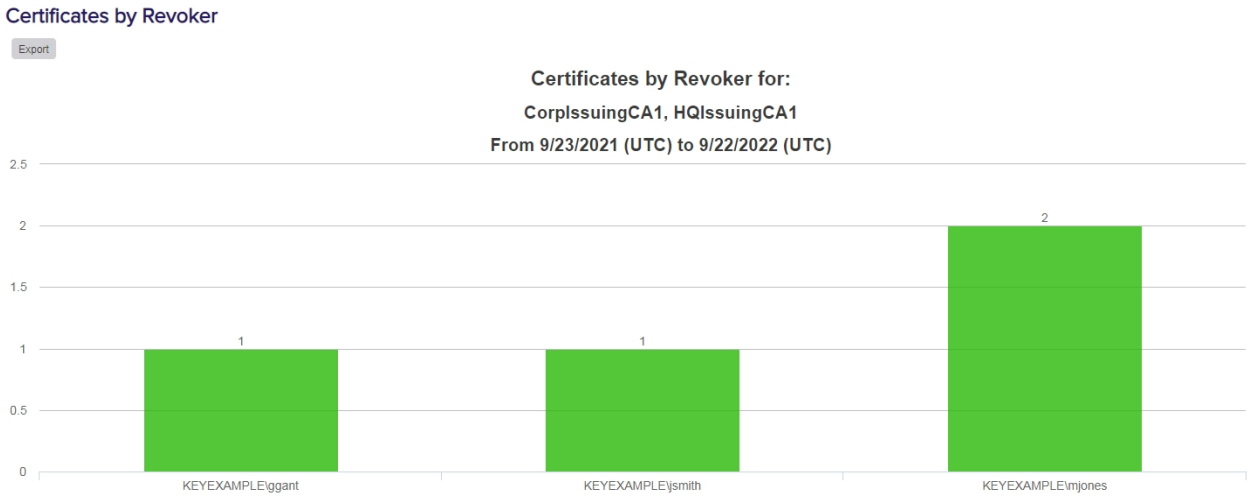


Figure 72: Certificates by Revoker

The export options for the Certificates by Revoker report are Excel and PDF.

Note: Certificates that have been revoked outside of Keyfactor Command (e.g. directly on the CA) appear with an *Unknown* revoker.

The input parameters for this report are:

- The evaluation date for the report. This report covers a specified number of days, weeks or months ending with this date. The default evaluation date is the current date, meaning

certificates revoked up to the current date will be included in the report. The default can be changed in the report parameters (see Report Manager Operations on page 127).

- The number of periods to include in the report. This is how many days, weeks or months of data to include in the report. The default is 52.
- The period length for the report. The options are days, weeks or months. The default is weeks.
- The CA(s) to include in the report. Certificates that were issued from CA(s) other than those selected will not be included in the counts of revoked certificates.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

Note: Only CAs currently or previously configured for synchronization are available for reporting.

2.1.4.7 Certificates by Type and Java Keystore

The Certificates by Type and Java Keystore report provides a table with a summary of the number of certificates generated through Keyfactor Command in the selected date range broken down by PFX requests versus CSR requests for a selected CA or CAs. In addition, a count is provided of certificates that were added to Java Keystores in this timeframe (new or existing certificates from any source).

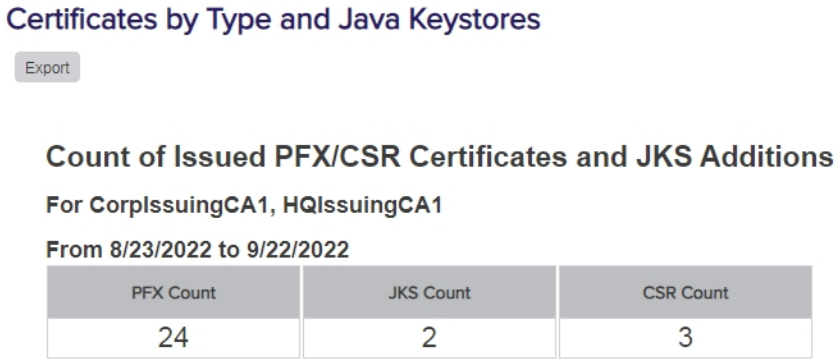


Figure 73: Certificates by Type and Java Keystore

The export options for the Certificates by Type and Java Keystore report are Excel and PDF.

The input parameters for this report are:

- The CA(s) to include in the report. Certificates that were issued from CAs other than those selected will not be included in the counts of PFXs and CSRs.
- The start date and end date for the date range on which to report. The default date range is one month ending with the current date. These defaults can be changed in the report parameters

(seeReport Manager Operations on page 127).

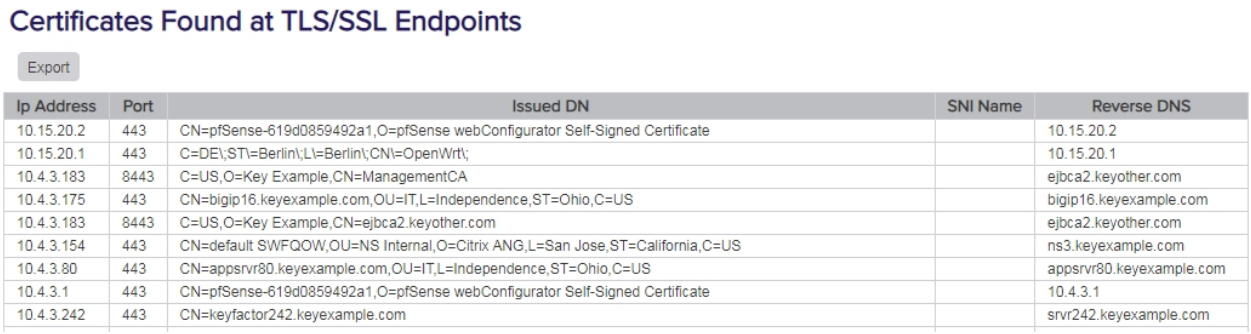
Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

Note: Only CAs currently or previously configured for synchronization are available for reporting.

2.1.4.8 Certificates Found at TLS/SSL Endpoints

The Certificates Found at TLS/SSL Endpoints report provides a table which shows the IP Address and Port for any discovered endpoints with the Issued DN for the certificate or certificates discovered at the endpoint and the server name indication (SNI) configured on the endpoint, if available. The report table includes these fields:

- IP Address
- Port
- Issued DN
- SNI Name
- Reverse DNS



Ip Address	Port	Issued DN	SNI Name	Reverse DNS
10.15.20.2	443	CN=pfSense-619d0859492a1,O=pfSense webConfigurator Self-Signed Certificate		10.15.20.2
10.15.20.1	443	C=DE\;ST\=Berlin\;L\=Berlin\;CN\=OpenWrt\;		10.15.20.1
10.4.3.183	8443	C=US,O=Key Example,CN=ManagementCA		ejbca2.keyother.com
10.4.3.175	443	CN=bigip16.keyexample.com,OU=IT,L=Independence,ST=Ohio,C=US		bigip16.keyexample.com
10.4.3.183	8443	C=US,O=Key Example,CN=ejbca2.keyother.com		ejbca2.keyother.com
10.4.3.154	443	CN=default SWFQOW,OU=NS Internal,O=Citrix ANG,L=San Jose,ST=California,C=US		ns3.keyexample.com
10.4.3.80	443	CN=appsrvr80.keyexample.com,OU=IT,L=Independence,ST=Ohio,C=US		appsrvr80.keyexample.com
10.4.3.1	443	CN=pfSense-619d0859492a1,O=pfSense webConfigurator Self-Signed Certificate		10.4.3.1
10.4.3.242	443	CN=keyfactor242.keyexample.com		srvr242.keyexample.com

Figure 74: Certificates Found at TLS/SSL Endpoints

The export options for the Certificates Found at TLS/SSL Endpoints report are CSV and Excel.

The input parameters for this report are:

- The orchestrator pool on which to report. Only one orchestrator pool can be selected. A default for the orchestrator pool on which to report can be configured in the report parameters (see Report Manager Operations on page 127).
- The start date and end date for the date range on which to report. The default date range is one month ending with the current date. These defaults can be changed in the report parameters (see Report Manager Operations on page 127).

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

2.1.4.9 Certificates in Collection

The Certificates in Collection report shows detailed information for the active, expired and revoked certificates in the selected collection.

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

The export options for the Certificates in Collection report are CSV and Excel.

The report table includes these fields:

- ID

  The Keyfactor Command reference ID for the certificate.
- Issued DN
- Effective Date (UTC)
- Expiration Date (UTC)
- Issued CN
- Issuer DN
- Principal

  The user principal name (UPN) contained in the subject alternative name (SAN) field of the certificate, if present (e.g. username@keyexample.com).
- Requester
- Thumbprint
- Template
- Cert State

  The state of the certificate (e.g. Active, Revoked, Unknown).
- Key Type
- Key Size in Bits
- Key Usage
- Signing Algorithm
- Serial Number
- CA Record ID

  The ID of the certificate in the CA database.
- Issued OU

  The OU from the certificate subject, if any.
- Issued Email

  The email address from the certificate subject, if any.
- Revocation Effective Date
- Revocation Reason
- Metadata (Optional)

Column handling on this report grid has the following features:

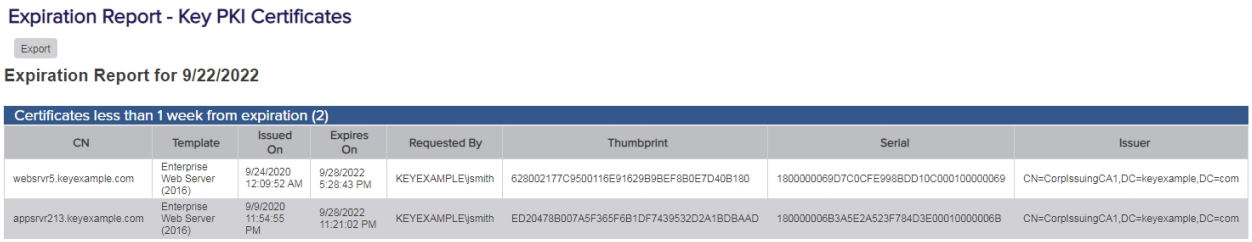
- To change the width of a column of the report, hover over the triangle of dots on the right side of the selected column header ( ). Click, hold and drag the triangle to change the width of the column.
- To rearrange columns on the report display, hover over the rectangle of dots on the left side of the selected column header ( ). Click, hold, and drag the rectangle to move the column to your selected location.
- Most columns can be sorted in ascending order by clicking on the header of the column. Click the column header again to reverse the sort order. When a column is sorted, a caret will appear at the end of the column name showing the direction of the sort. Lack of a triangle indicates the report is sorted by the default column and order.

The input parameter for this report is:

- The certificate collection to report on, including the built-in option, *All Certificate*s collection. The default is *All Certificates*.
- The metadata field(s) to include, if desired.

2.1.4.10 Expiration Report

The Expiration Report includes table(s) showing detailed information for certificates expiring and expired within the next 12 weeks and CA certificates expiring and expired within the next 2 1/2 years. Expired certificates are only included if they have expired within the last 4 weeks.



Expiration Report - Key PKI Certificates

Export

Expiration Report for 9/22/2022

Certificates less than 1 week from expiration (2)

CN	Template	Issued On	Expires On	Requested By	Thumbprint	Serial	Issuer
websrvr5.keyexample.com	Enterprise Web Server (2016)	9/24/2020 12:09:52 AM	9/28/2022 5:28:43 PM	KEYEXAMPLE\jsmith	628002177C9500116E91629B9BEF8B0E7D40B180	1800000069D7C0CFE998BDD10C000100000069	CN=CorpIssuingCA1,DC=keyexample,DC=com
appsrvr213.keyexample.com	Enterprise Web Server (2016)	9/9/2020 11:54:55 PM	9/28/2022 11:21:02 PM	KEYEXAMPLE\jsmith	ED20478B007A5F365F6B1DF7439532D2A1BDBAAD	180000006B3A5E2A523F784D3E00010000006B	CN=CorpIssuingCA1,DC=keyexample,DC=com

Figure 75: Certificate Expiration Report: Certificates Expiring within One Week

The export options for the Expiration report are Excel and PDF. The PDF exports in landscape format to accommodate the wide width of the report.

The report includes the following tables:

- Expired Certificates (within the last 4 weeks)
- Certificates less than 1 week from expiration
- Certificates less than 2 weeks from expiration
- Certificates less than 4 weeks from expiration
- Certificates less than 6 weeks from expiration
- Certificates less than 8 weeks from expiration
- Certificates less than 12 weeks from expiration

In addition, tables are shown for CA certificates expiring in the following timeframes relative to the selected report date:

- CA certificates less than 6 months from expiration
- CA certificates less than 12 months from expiration
- CA certificates less than 18 months from expiration
- CA certificates less than 24 months from expiration
- CA certificates less than 30 months from expiration

A table is only shown if a certificate or CA in the collection matches the expiration time window. A certificate or CA appears in only one table, so, for example, a certificate expiring within 4 weeks does not also appear as expiring within 6 weeks.

The report tables include these fields:

- CN (Common Name)
- Template
- Issued On
- Expires On (this is the default sort order)
- Requested By
- Thumbprint
- Serial (Number)
- Issuer (Distinguished Name)
- Metadata (optional)

Column handling on this report grid has the following features:

- To change the width of a column of the report, hover over the triangle of dots on the right side of the selected column header (). Click, hold and drag the triangle to change the width of the column.
- To rearrange columns on the report display, hover over the rectangle of dots on the left side of the selected column header (). Click, hold, and drag the rectangle to move the column to your selected location.
- Most columns can be sorted in ascending order by clicking on the header of the column. Click the column header again to reverse the sort order. When a column is sorted, a caret will appear at the end of the column name showing the direction of the sort. Lack of a triangle indicates the report is sorted by the default column and order.

The input parameters for this report are:

- The certificate collection to report on, including the built-in *All Certificates* collection. The default is *All Certificates*.
- The evaluation date to report on. The default is the current date.
- The metadata field(s) to include, if desired.

Tip: This report makes use of the optional certificate de-duplication logic by default. When de-duplication is enabled for a report, the report results will include only the most recently issued certificate if there is more than one certificate that matches the de-duplication criteria. De-duplication is enabled for a report by checking the *Ignore Renewed Certificates* box on the Details tab of the report configuration (see Report Manager Operations on page 127). De-duplication can only be enabled for reports that use certificate collections—the *Uses Collection* box on the Details tab. The *Uses Collection* setting is not user-configurable.

De-duping is configured on a certificate collection by setting the *Ignore renewed certificate results by* option when saving a certificate collection (see Saving Search Criteria as a Collection on page 42). Certificate collections may be configured to be de-duplicated based on the certificate common name, distinguished name, or principal name (or not at all). Only certificates that share all the EKUs (e.g. Client Authentication and Server Authentication) as well as the same CN, DN or UPN will be eliminated as duplicates. If a certificate has more than one EKU and at least one EKU does not match an otherwise similar certificate with matching CN, DN or UPN, it will not be eliminated.

For example, if the de-duplication logic was set to DN and the report would include these two certificates:

- Certificate one:
  - DN: CN=apps-srvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication
  - Issued Date: December 1, 2020
  - Expiration Date: January 1, 2022
- Certificate two:
  - DN: CN=apps-srvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication
  - Issued Date: December 15, 2020
  - Expiration Date: December 14, 2021

The de-duplication logic would be triggered because the DNs and EKUs match. The report would include certificate two and leave out certificate one. Notice that certificate two is retained even through certificate one expires after certificate two. This is because certificate two was issued after certificate one.

Now imagine that the de-duplication logic is set to CN and the report would include these two certificates:

- Certificate one:
  - DN: CN=appsrvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L-L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication
  - Issued Date: December 1, 2020
  - Expiration Date: January 1, 2022
- Certificate two:
  - DN: CN=appsrvr14.keyexample.com,OU=HR,O=Key Example, Inc.,L-L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication, Client Authentication
  - Issued Date: December 15, 2020
  - Expiration Date: December 14, 2021

Although the DNs for these certificates do not match, the CNs still do, so this matches the de-duplication logic of CN. However, the EKUs for these two certificates do not match, since only one of them includes Client Authentication. In this case, both certificates would appear on the report.

Note: This report is limited to a maximum of 10,000 expiring and recently expired (within the last 4 weeks) certificates on which to report. Selecting a certificate collection containing more expiring and recently expired certificates than this, based on the evaluation date, will result in an error. Selecting a certificate collection containing a large number of certificates to report on can cause the report to take a long time to generate.

2.1.4.11 Expiration Report by Days

The Expiration Report by Days shows details for certificates expiring after a given start date with a time span chosen in days. It can be used, for example, to show you all the certificates in a certificate collection expiring within the next few days.

The Expiration Report includes a table showing detailed information for certificates expiring in the time frames identified by the parameters start date and number of days. The number of days parameter value must be between 0 and 100.

The export options for the Expiration Report by Days are CSV and Excel.

The report tables include these fields:

- CN (Common Name)
- Template
- Issued On
- Expires On (this is the default sort order)
- Requested By
- Thumbprint
- Serial (Number)
- Issuer (Distinguished Name)
- Metadata (optional)

Column handling on this report grid has the following features:

- To change the width of a column of the report, hover over the triangle of dots on the right side of the selected column header (). Click, hold and drag the triangle to change the width of the column.
- To rearrange columns on the report display, hover over the rectangle of dots on the left side of the selected column header (). Click, hold, and drag the rectangle to move the column to your selected location.

- Most columns can be sorted in ascending order by clicking on the header of the column. Click the column header again to reverse the sort order.

The input parameters for this report are:

- The certificate collection to report on, including the built-in *All Certificates* collection. The default is *All Certificates*.
- The start date of the reporting period. The default is the current date.
- The number of days in the reporting period (must be between 0 and 100). The default is 6.
- The metadata field(s) to include, if desired.

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

Tip: This report makes use of the optional certificate de-duplication logic by default. When de-duplication is enabled for a report, the report results will include only the most recently issued certificate if there is more than one certificate that matches the de-duplication criteria. De-duplication is enabled for a report by checking the *Ignore Renewed Certificates* box on the Details tab of the report configuration (see Report Manager Operations on page 127). De-duplication can only be enabled for reports that use certificate collections—the *Uses Collection* box on the Details tab. The *Uses Collection* setting is not user-configurable.
De-duping is configured on a certificate collection by setting the *Ignore renewed certificate results by* option when saving a certificate collection (see Saving Search Criteria as a Collection on page 42). Certificate collections may be configured to be de-duplicated based on the certificate common name, distinguished name, or principal name (or not at all). Only certificates that share all the EKUs (e.g. Client Authentication and Server Authentication) as well as the same CN, DN or UPN will be eliminated as duplicates. If a certificate has more than one EKU and at least one EKU does not match an otherwise similar certificate with matching CN, DN or UPN, it will not be eliminated.

For example, if the de-duplication logic was set to DN and the report would include these two certificates:

- Certificate one:
  - DN: CN=apps-srvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication
  - Issued Date: December 1, 2020
  - Expiration Date: January 1, 2022
- Certificate two:
  - DN: CN=apps-srvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication
  - Issued Date: December 15, 2020
  - Expiration Date: December 14, 2021

The de-duplication logic would be triggered because the DNs and EKUs match. The report would include certificate two and leave out certificate one. Notice that certificate two is retained even through certificate one expires after certificate two. This is because certificate two was issued after certificate one.

Now imagine that the de-duplication logic is set to CN and the report would include these two certificates:

- Certificate one:
  - DN: CN=appsrvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L-L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication
  - Issued Date: December 1, 2020
  - Expiration Date: January 1, 2022
- Certificate two:
  - DN: CN=appsrvr14.keyexample.com,OU=HR,O=Key Example, Inc.,L-L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication, Client Authentication
  - Issued Date: December 15, 2020
  - Expiration Date: December 14, 2021

Although the DNs for these certificates do not match, the CNs still do, so this matches the de-duplication logic of CN. However, the EKUs for these two certificates do not match, since only one of them includes Client Authentication. In this case, both certificates would appear on the report.

Note: This report is limited to a maximum of 10,000 expiring certificates on which to report. Selecting a certificate collection containing more expiring certificates than this, within the selected reporting period, will result in an error. Selecting a certificate collection containing a large number of certificates to report on can cause the report to take a long time to generate.

2.1.4.12 Full Certificate Extract Report

The Full Certificate Extract Report shows detailed information for the active, expired and revoked certificates in the selected collection.

The export options for the Full Certificate Extract Report are CSV and Excel.

The report table includes these fields:

- Common Name
  The common name of the certificate.
- Valid From
  The date on which the certificate became valid (typically the issuance date).
- Valid To
  The date on which the certificate expires.
- Total SANs
  The total number of subject alternative names (SANs) for the certificate.
- SANs
  Any subject alternative names (SANs) of type DNS name, UPN, or email.
- SANs IP

- Days to Expiration

  The number of days remaining until the certificate expires. This will be a negative value for expired certificates.
- Signature Algorithm

  The cryptographic algorithm used to sign the certificate.
- Key Size

  The key length used to create the certificate.
- Validity Period

  The number of days for which the certificate was issued.
- Serial Number

  The serial number of the certificate.
- DN

  The distinguished name (subject) of the certificate.
- Issuer DN

  The distinguished name of the issuer (CA) for the certificate.
- User Name

  The name of the identity that requested the certificate.

  Any subject alternative names (SANs) of type IP address.
- Port

  The port where the certificate was found on an SSL scan.
- IP Address

  The IP address where the certificate was found on an SSL scan.
- DNS Name

  The DNS name resolved for the IP address where the certificate was found on an SSL scan.
- Alias

  The alias of the certificate in the certificate store.
- Client Machine

  Depending on the type of certificate store, either the name of the server on which the orchestrator is installed or the name of the server on which the certificate store is located.
- Store Path

  The location of the certificate store. The format of this value will vary depending on the type of certificate store.
- Template

  The certificate template used to issue the certificate.

Column handling on this report grid has the following features:

- To change the width of a column of the report, hover over the triangle of dots on the right side of the selected column header (). Click, hold and drag the triangle to change the width of the column.
- To rearrange columns on the report display, hover over the rectangle of dots on the left side of the selected column header (). Click, hold, and drag the rectangle to move the column to your selected location.
- Most columns can be sorted in ascending order by clicking on the header of the column. Click the column header again to reverse the sort order.

This report takes the input parameters:

- The certificate collection to report on, including the built-in option, *All Certificates* collection. The default is *All Certificates*.
- The metadata field(s) to include, if desired. This will append the selected metadata columns to the end of the report.

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

2.1.4.13 Issued Certificates per Certificate Authority

The Issued Certificates per Certificate Authority report includes line graphs showing the number of certificates issued for each template in the selected date range for the selected template(s) on the selected CA. A separate line graph is generated for each template. An option to report on certificates that are not associated with any CA is included.

Note: Clicking on a bar on the graph, or a section of a pie chart or line graph, will open a new window as a drill down to the certificate search grid filtered for the exact criteria of that aspect of the graph. You can return to the original report by navigating to the original report window.

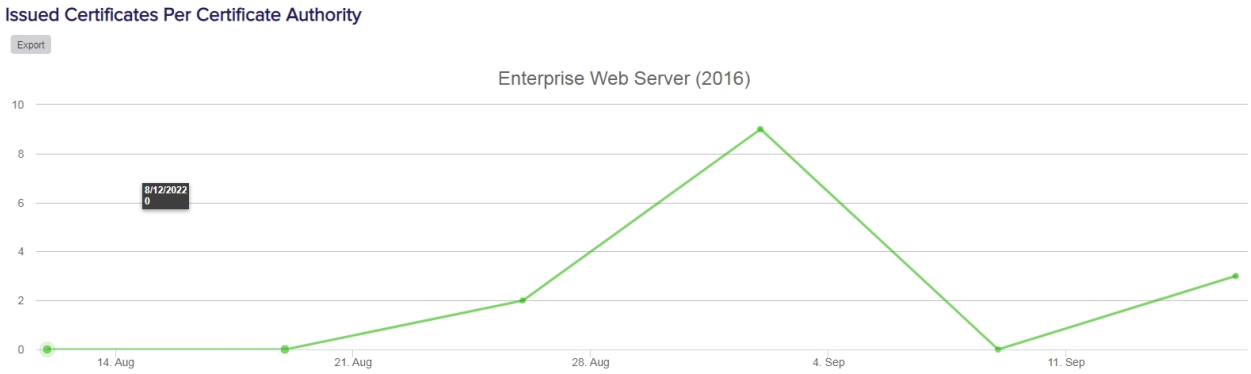


Figure 76: Issued Certificates per CA

The export options for the Issued Certificates per Certificate Authority report are Excel and PDF.

The input parameters for this report are:

- The evaluation date for the report. This report covers a specified number of days, weeks or months ending with this date. The default evaluation date is the current date, meaning certificates issued up to the current date will be included in the report.
- The number of periods to include in the report. This is how many days, weeks or months of data to include in the report. The default is 6.
- The period length for the report. The options are days, weeks or months. The default is weeks.

- Which CA to include in the report. This includes the option to report on certificates that have no associated CA. Typically, these would be certificates found via SSL scanning or inventory on certificate stores. Only one CA option can be reported on at a time.
- The template(s) to include in the report. A separate line graph is generated for each template selected for reporting. Templates that are available for issuance from more than one CA are reported separately by CA, so only certificates issued for the selected template *and* the selected CA will be shown. When the *Certificates Not Associated with CA* option is selected for the CA, the *No Template* option should be selected for the template.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

Note: Other than the option of certificates with no associated CA, only CAs currently configured for synchronization are available for reporting.

2.1.4.14 Monthly Executive Report

The Monthly Executive report provides a dashboard-like summary including bar and pie charts with counts of certificates created, renewed and approaching expiration for a selected CA or CAs. Data for certificates approaching expiration is presented in a pie chart broken out into certificates that will expire in the next 15 days, in 16-30 days, 31-60 days and 61-90 days. Data for certificates that have been recently created or renewed is presented in a bar chart that includes data for the current month and the previous month, broken out by month and renewed versus newly created. In addition, a summary pie chart is included that shows all the active certificates for the selected CAs broken out by CA.

Note: Clicking on a bar on the graph, or a section of a pie chart or line graph, will open a new window as a drill down to the certificate search grid filtered for the exact criteria of that aspect of the graph. You can return to the original report by navigating to the original report window.

The export options for the Monthly Executive report are Excel and PDF.

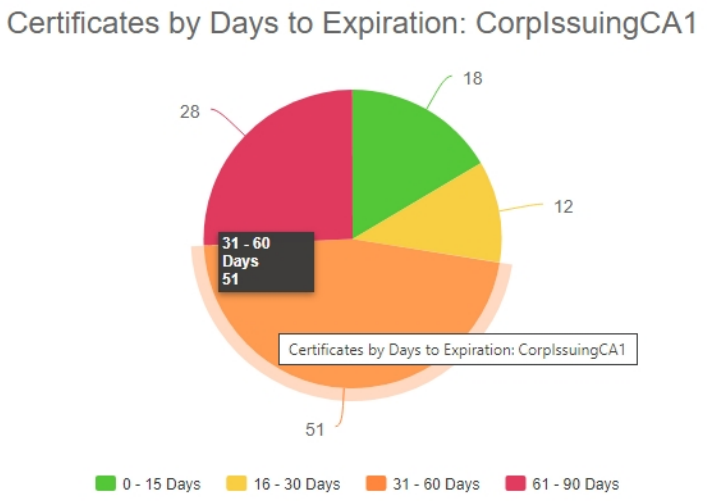


Figure 77: Example Pie Chart from Monthly Executive Report

This report takes as an input parameter the CA or CAs to report on and includes the option to report on certificates that have no associated CA. Typically, these would be certificates found via SSL scanning or inventory on certificate stores.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

Note: Only CAs currently or previously configured for synchronization are available for reporting.

2.1.4.15 PKI Status for Collection

The PKI Status for Collection report is a multi-page report incorporating tables and charts that provides an overview of the status of the certificates in the selected collection.

Note: Clicking on a bar on the graph, or a section of a pie chart or line graph, will open a new window as a drill down to the certificate search grid filtered for the exact criteria of that aspect of the graph. You can return to the original report by navigating to the original report window.

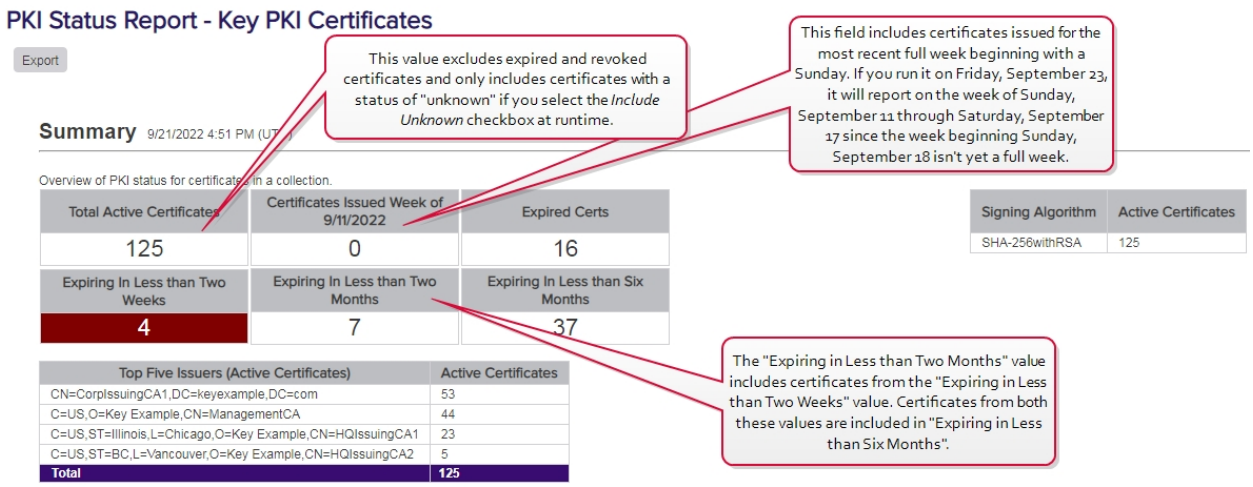


Figure 78: PKI Status for Collection Summary

The export options for the PKI Status for Collection report are Excel and PDF.

This report takes as an input parameter the certificate collection to report on, including the built-in *All Certificates* collection, and has the option to include or exclude certificates that have a status of unknown (certificates found on SSL scans and in certificate stores often have this status). The default collection is *All Certificates*, and unknown certificates are excluded by default.

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

Sections of the report include:

Summary Page

The summary page provides certificate counts for the following:

- Total number of active certificates

  This value excludes expired and revoked certificates and only includes non-expired, non-revoked certificates with an unknown state if the *Include Unknown* checkbox is selected at runtime.
- Number of certificates issued in the most recently completed week, beginning with a Sunday
- Number of expired certificates
- Number of certificates coming up for expiration within two weeks

- Number of certificates coming up for expiration within two months (including those expiring within two weeks)
- Number of certificates coming up for expiration within six months (including those expiring within two weeks and two months)
- A breakdown of the number of active certificates by signing algorithm (only the top five signing algorithms are shown)
- The top five issuers of active certificates with the number of active certificates

Next Ten Certificates to Expire Page

This table shows details of the ten certificates expiring within the shortest timeframe (for any time-frame under two years) and includes the certificate CN, issuer CN, certificate validity period in UTC time, template name, thumbprint and serial number. Grid columns may be rearranged by click-holding and dragging the grid arrangement control icon ( ) at the left of the column header. Click a column header to sort the grid by ascending values; click again to sort descending (columns already in ascending order will switch to descending on the first click). The screen will redraw when you sort. Not all columns are sortable.

PKI Health Metrics—Lifetime Remaining Page

This donut chart shows the percentage of certificates that are expired, near expiration (90% or more of lifetime used) or active and not near expiration along with a table showing the specific numbers in these categories. Hover over a segment in the donut chart to see details for that segment. Click one of the labels below the donut chart to toggle add/remove the segment on the chart. Grid columns may be rearranged by click-holding and dragging the grid arrangement control icon ( ) at the left of the column header. Click a column header to sort the grid by ascending values; click again to sort descending (columns already in ascending order will switch to descending on the first click). The screen will redraw when you sort. Not all columns are sortable.

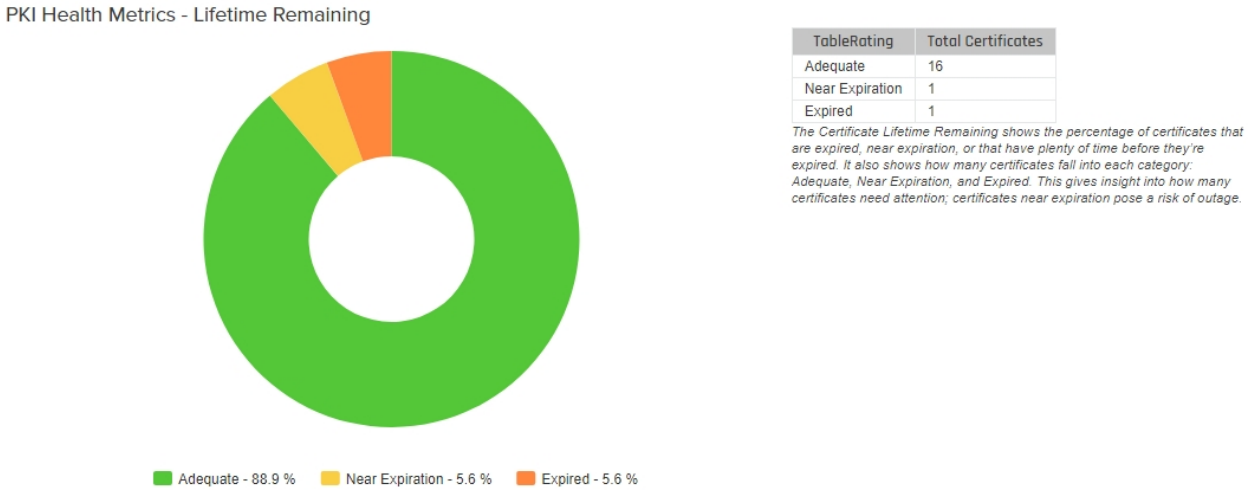


Figure 79: PKI Status for Collection Lifetime Remaining

PKI Health Metrics—Algorithm Strength

This donut chart shows the percentage of certificates (active and expired) with strong (SHA2 and SSA), weak (SHA1) or critically weak (MD5 and older) signature algorithms along with a table showing the specific numbers in these categories. Hover over a segment in the donut chart to see details for that segment. Click one of the labels below the donut chart to toggle add/remove the segment on the chart. Grid columns may be rearranged by click-holding and dragging the grid arrangement control icon () at the left of the column header. Click a column header to sort the grid by ascending values; click again to sort descending (columns already in ascending order will switch to descending on the first click). The screen will redraw when you sort. Not all columns are sortable.

PKI Health Metrics—RSA Key Strength

This donut chart shows the percentage of certificates (active and expired) with strong (2048+), weak (1024-2047), and critically weak (<1024) RSA keys along with a table showing the specific numbers in these categories. Hover over a segment in the donut chart to see details for that segment. Click one of the labels below the donut chart to toggle add/remove the segment on the chart. Grid columns may be rearranged by click-holding and dragging the grid arrangement control icon () at the left of the column header. Click a column header to sort the grid by ascending values; click again to sort descending (columns already in ascending order will switch to descending on the first click). The screen will redraw when you sort. Not all columns are sortable.

Certificates by Signing Algorithm

This donut chart shows the percentage of active certificates broken down by signing algorithm (RSA SHA-1, RSA SHA-256, etc.) along with a table showing the specific numbers in these categories. Hover over a segment in the donut chart to see details for that segment. Click one of the labels below the donut chart to toggle add/remove the segment on the chart.

Top Certificate Issuers

This donut chart shows the percentage of certificates (active and expired) broken down by the top five issuers plus an *other* bucket along with a table showing the specific numbers in these categories. Hover over a segment in the donut chart to see details for that segment. Click one of the labels below the donut chart to toggle add/remove the segment on the chart. Grid columns may be rearranged by click-holding and dragging the grid arrangement control icon () at the left of the column header. Click a column header to sort the grid by ascending values; click again to sort descending (columns already in ascending order will switch to descending on the first click). The screen will redraw when you sort. Not all columns are sortable.

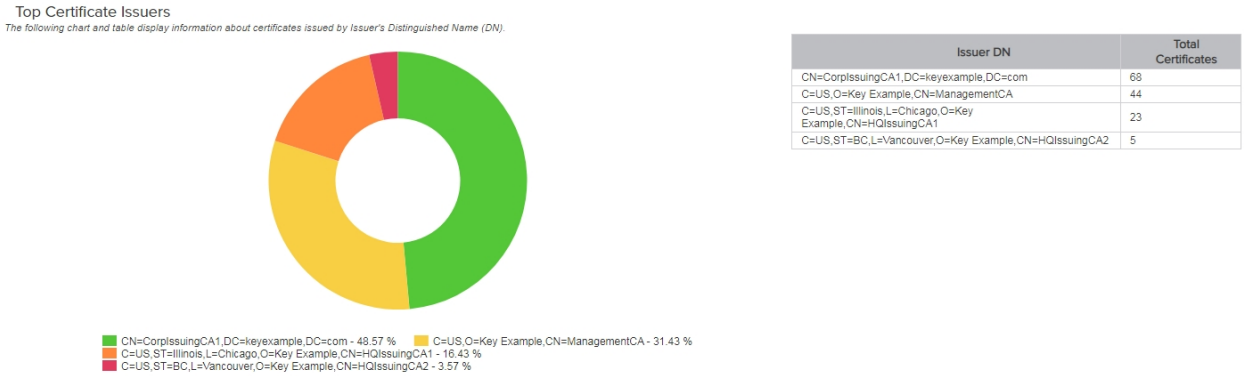


Figure 80: PKI Status for Collection Top Issuers

Certificates Issued in Previous 10 Weeks

This bar chart shows the number of certificates (active and expired) issued per week for the ten weeks leading up to and through the full week prior to the run date of the report. Hover over a bar to see the number of issued certificates for the week with that date.

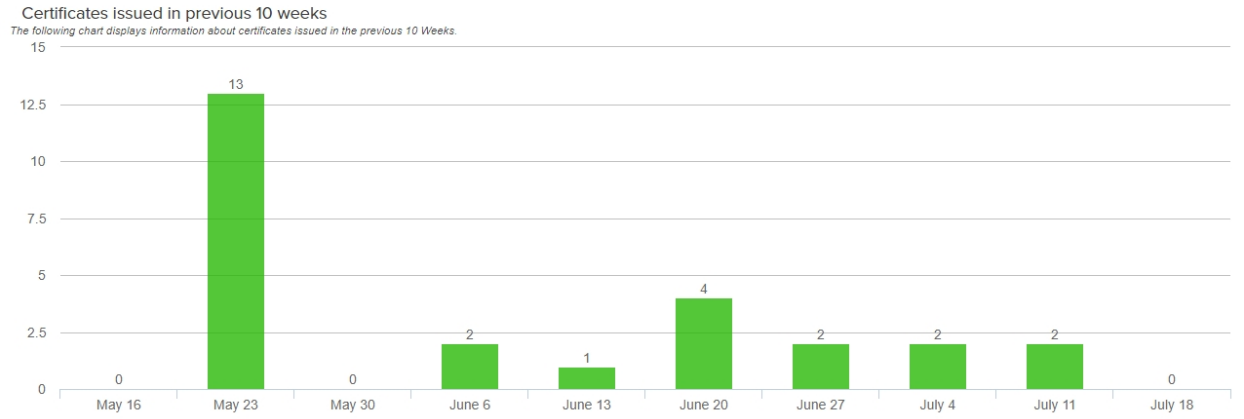


Figure 81: PKI Status for Certificates issued in previous 10 weeks

Certificates Issued in Previous 12 Months

This bar chart shows the number of certificates (active and expired) issued per month for the twelve months leading up to and through the full month prior to the run date of the report, broken down by internally issued certificates (from sources managed by Keyfactor Command such as synchronization of CAs in the primary forest and any trusted forests, any certificate vendors synced using a Keyfactor gateway, and any CAs synced using the remote CA agent) and externally issued certificates (from sources not managed by Keyfactor Command such as certificates located during SSL scans or uploaded using the Add Certificate option). Hover over a bar to see the number of issued certificates for that month and source. Click one of the labels below the chart to toggle add/remove the segment on the chart.

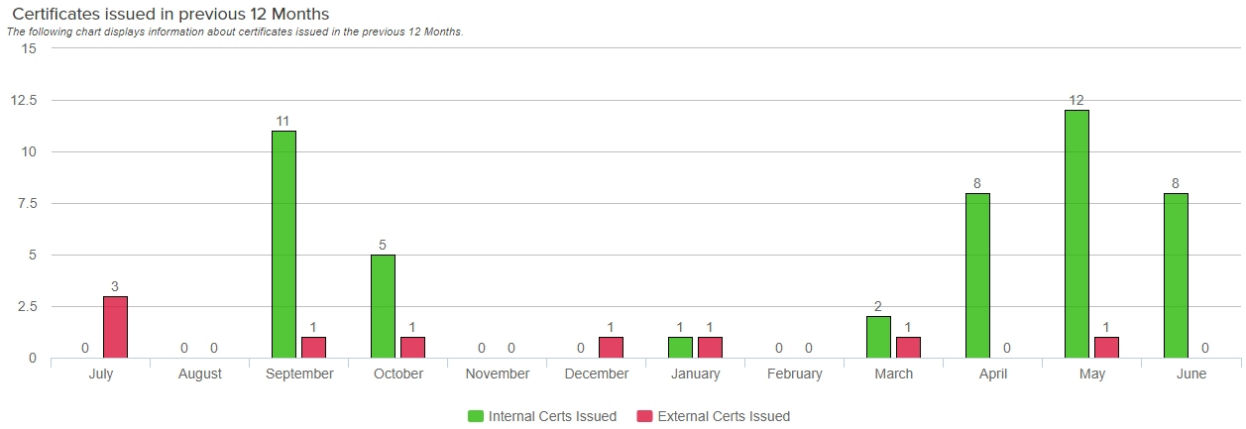


Figure 82: PKI Status for Certificates issued in previous 12 months

Weak RSA Certificates

This table shows details of the certificates with weak (under 2048) RSA keys and includes the certificate CN, issuer CN, certificate validity period in UTC time, key size, thumbprint and serial number. A maximum of 1000 certificates is shown. Grid columns may be rearranged by click-holding and dragging the grid arrangement control icon ( ) at the left of the column header. Click a column header to sort the grid by ascending values; click again to sort descending (columns already in ascending order will switch to descending on the first click). The screen will redraw when you sort. Not all columns are sortable.

Deprecated Signing Algorithms

This table shows details of the certificates with deprecated (MD5 and older) signing algorithms and includes the certificate CN, issuer CN, certificate validity period in UTC time, signing algorithm, thumbprint and serial number. A maximum of 1000 certificates is shown. Grid columns may be rearranged by click-holding and dragging the grid arrangement control icon ( ) at the left of the column header. Click a column header to sort the grid by ascending values; click again to sort descending (columns already in ascending order will switch to descending on the first click). The screen will redraw when you sort. Not all columns are sortable.

Self-Signed Certificates

This table shows details of the certificates that are self-signed or root CA certificates and includes the certificate DN, certificate validity period in UTC time, thumbprint and serial number. A maximum of 1000 certificates is shown. Grid columns may be rearranged by click-holding and dragging the grid arrangement control icon ( ) at the left of the column header. Click a column header to sort the grid by ascending values; click again to sort descending (columns already in ascending order will switch to descending on the first click). The screen will redraw when you sort. Not all columns are sortable.

2.1.4.16 Revoked Certificates in Certificate Stores

The Revoked Certificates in Certificate Stores report displays a table of all certificates that have been revoked, either in Keyfactor Command or externally, that are found in at least one certificate store or SSL scan location, and for which the revocation effective date is less than or equal to the date and time when the report is run (not in the future). The report is included in the report manager Certificate Locations and Certificate Lifecycle categories.

The export options for the Revoked Certificates in Certificate Stores report are CSV and Excel.

The report table includes these fields:

- Certificate CN

  The common name of the certificate.
- Thumbprint

  The thumbprint of the certificate.
- User

  The username (DOMAIN\username format) of the user who revoked the certificate.
- Expiration Date (UTC)

  The date on which the certificate expires.
- Issued Date (UTC)

  The date on which the certificate became valid (typically the issuance date).
- Template Name

  The certificate template used to issue the certificate.
- SSL Location

  The DNS name(s) resolved for the IP address(es) where the certificate was found on an SSL scan. Due to query constraints, the maximum length of text allowed in each of these fields is 10,000 characters.
- Cert Store Location

  The name(s) of the server(s) on which the certificate is found in one or more certificate stores and the location of the certificate store(s). The format of this value will vary depending on the type of certificate store. Due to query constraints, the maximum length of text allowed in each of these fields is 10,000 characters.
- Revocation Date (UTC)

  The date on which the certificate was revoked in UTC.
- Revocation Reason

  The reason given for the certificate revocation.
- Revocation Comment

  The comment entered at revocation.

Column handling on this report grid has the following features:

- To change the width of a column of the report, hover over the triangle of dots on the right side of the selected column header (). Click, hold and drag the triangle to change the width of the column.
- To rearrange columns on the report display, hover over the rectangle of dots on the left side of the selected column header (). Click, hold, and drag the rectangle to move the column to your selected location.
- Most columns can be sorted in ascending order by clicking on the header of the column. Click the column header again to reverse the sort order. When a column is sorted, a caret will appear at the end of the column name showing the direction of the sort. Lack of a triangle indicates the report is sorted by the default column and order.

This report takes as an input parameter the certificate collection to report on, including the built-in *All Certificates* collection. The default is *All Certificates*.

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

Note: This report is limited to a maximum of 100,000 revoked certificates in certificate stores on which to report. Selecting a certificate collection containing more certificates than this will result in an error.

2.1.4.17 SSH Key Usage

The SSH Key Usage report shows a table which displays a list of SSH keys that have not been used to log on in the given minimum number of days.

The export options for the SSH Key Usage report are CSV and Excel.

The grid includes:

- Key Fingerprint
  The fingerprint of the SSH public key.
- Discovered Date
  The date and time (in local server time) on which the SSH key was discovered.
- Date Last Used
  The date and time (in local server time) on which the SSH key was last used.
- Key Length
  The key length of the SSH public key.
- Logon Username
  The Linux logon username associated with the key.
- Logon Server
  The IP address of the Linux server last used to logon.

This report takes as an input parameter; number of *Days Since Last Used*. You must select a number between 0 and 100.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

2.1.4.18 SSH Keys by Age

The SSH Keys by Age report shows one or more table(s) with detailed information for SSH keys generated in Keyfactor Command broken down by age—as defined by the *Key Lifetime (days)* application setting (see Application Settings: SSH Tab on page 620).

The export options for the SSH Keys by Age report are PDF and Excel.

The report aging categories are:

- Stale keys (within the last 4 weeks)
- Keys less than 1 week from being stale
- Keys less than 4 weeks from being stale
- Keys less than 8 weeks from being stale
- Keys less than 6 months from being stale
- Keys less than 12 months from being stale

A table is only shown if an SSH key with one of the selected key types matches the age window. An SSH key appears only in one table, so, for example, a key that will become stale within 4 weeks and appears in the 4-week table does not also appear as becoming stale within the 8-week table.

The grid includes:

- Account Name
  For user keys, the Active Directory user account associated with the key being reported on. For service account keys, the username and client hostname entered when the service account key was created (e.g. myapp@appsrvr.keyexample.com).
- Creation Date
  The date (in UTC time) on which the SSH key was created.
- Fingerprint
  The fingerprint of the SSH public key.
- Key Type
  The key type of the SSH public key.
- Key Length
  The key length of the SSH public key.
- Associated Logons
  The number of Linux logons associated with the SSH public key.

This report takes as an input parameter the SSH Key Types to include in the report. You must select at least one key type using the **Select SSH Key Types** button.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

2.1.4.19 SSH Keys with Root Logon Access

The SSH Keys with Root Logon Access report shows a list of SSH public keys found associated with root logon authorized_keys files on servers managed with the SSH orchestrator. Holders of the matching private keys for these public keys can gain root access without providing the root password.

The export options for the SSH Keys with Root Logon Access report are CSV and Excel.

The grid includes the fields:

- Account Name
  The Active Directory user account associated with the key found to have root access on the target machine, if any. This field will only be populated for keys created in Keyfactor Command.
- Fingerprint
  The fingerprint of the SSH public key found associated with the root logon on the target machine.
- Hostname
  The host name of the server on which the root logon was found to have an SSH public key providing logon access.
- Creation Date
  The date (in UTC time) on which the SSH key was created. This field will only be populated for keys created in Keyfactor Command.
- Date Found
  The date (in UTC time) on which Keyfactor Command found the root logon SSH public key on the target server. This field will only be populated for keys discovered outside of Keyfactor Command (as opposed to created in Keyfactor Command).
- Key Type
  The key type of the SSH public key found to have root access on the target machine.
- Key Length
  The key length of the SSH public key found to have root access on the target machine.

The input parameter for this report is:

- The start date and end date range for the report. This is the date range during which SSH keys that allow root logon were created or discovered by Keyfactor Command. The default start date is one month prior to the current date. The default end date is the current date, meaning only SSH keys with root access discovered or created within the last month will be included in the report.
- The SSH Key Types to include in the report.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

2.1.4.20 SSH Trusted Public Keys with No Known Private Keys

The SSH Trusted Public Keys with No Known Private Keys report shows a list of SSH public keys found in authorized_keys files on servers managed with the SSH orchestrator that do not have a matching private key in Keyfactor Command.

The export options for the SSH Trusted Public Keys with No Known Private Keys report are CSV and Excel.

The grid includes:

- Logon Name
  The Linux user account associated with the SSH public key found on the target machine.
- Fingerprint
  The fingerprint of the SSH public key found associated with the referenced logon on the target machine.
- Date Found
  The date (in UTC time) on which Keyfactor Command found the SSH public key on the target machine.
- Key Type
  The key type of the SSH public key found on the target machine.
- Key Length
  The key length of the SSH public key found on the target machine.
- Hostname
  The host name of the server on which the root logon was found to have an SSH public key providing logon access.
- Server Group
  The server group to which the server on which the root logon was found belongs.

The input parameters for this report are:

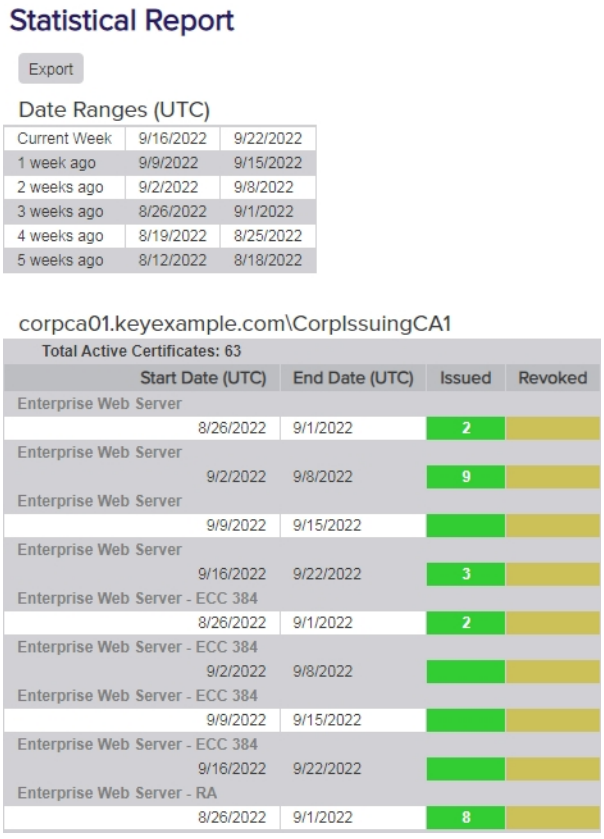
- The start date and end date range for the report. This is the date range during which SSH keys were discovered by Keyfactor Command. The default start date is one month prior to the current date. The default end date is the current date, meaning only SSH keys that have no matching private key discovered within the last month will be included in the report.
- The SSH Key Types to include in the report. You must select at least one key type using the **Select SSH Key Types** button.

Note: By default, this report is configured not to appear on the top menu under Reports and can be found only in Report Manager. You can change this by modifying the *Show in Navigator* setting (see Report Manager Operations on page 127).

2.1.4.21 Statistical Report

The Statistical Report shows the number of issued, revoked, and failed (includes denied) certificates for a user-definable period of time leading up to a user-definable date broken down by CA, certificate template and reporting period length (day, week or month). The report includes sections titled "No Template Associated" for certificates with no associated template. This may be the case with certificates issued from a standalone CA as well as select failed certificate requests from enterprise CAs.

The export options for the Statistical Report are PDF and Excel.


Statistical Report

Export

Date Ranges (UTC)

Current Week	9/16/2022	9/22/2022
1 week ago	9/9/2022	9/15/2022
2 weeks ago	9/2/2022	9/8/2022
3 weeks ago	8/26/2022	9/1/2022
4 weeks ago	8/19/2022	8/25/2022
5 weeks ago	8/12/2022	8/18/2022

corpca01.keyexample.com\CorpIssuingCA1

Total Active Certificates: 63

	Start Date (UTC)	End Date (UTC)	Issued	Revoked
Enterprise Web Server				
	8/26/2022	9/1/2022	2	
Enterprise Web Server				
	9/2/2022	9/8/2022	9	
Enterprise Web Server				
	9/9/2022	9/15/2022		
Enterprise Web Server				
	9/16/2022	9/22/2022	3	
Enterprise Web Server - ECC 384				
	8/26/2022	9/1/2022	2	
Enterprise Web Server - ECC 384				
	9/2/2022	9/8/2022		
Enterprise Web Server - ECC 384				
	9/9/2022	9/15/2022		
Enterprise Web Server - ECC 384				
	9/16/2022	9/22/2022		
Enterprise Web Server - RA				
	8/26/2022	9/1/2022	8	

Figure 83: Example Portion of the Statistical Report

The input parameters for this report are:

- The evaluation date of the reporting period. The default is the current date.
- The number of periods to report on. The default is six.

  Note: A maximum of 100 periods may be selected (e.g. 100 weeks).

- The period length—day, week or month. The default is week.

Tip: The *Total Active Certificates* count for each CA section in the report includes all certificates issued by that CA which are still active (not revoked and not expired), not just those issued in the time period for the report.

2.1.4.22 Report Manager

The Report Manager is used to run reports and manage existing reports, including scheduling delivery of reports. The Report Manager page shows all the available reports, not just those that have been configured to appear on the Management Portal top menu under *Reports*. Built-In Reports and Custom Reports are shown on separate tabs on the Report Manager page. Built-In reports have been organized into categories to allow you to filter the search results on the Report Manager grid by category of report.

With the Report Manager, custom Logi Analytics reports or custom reports from other external reporting solutions can be added into the portal to allow for easy running and scheduling. If you would like assistance creating a custom report in the new reporting engine, Logi Analytics, or displaying a custom report in the Report Manager, please contact your Client Success representative.

Tip: Be sure to check the filter on the category if you are not seeing all of the reports you expect to see. The default filter is *All* unless you have favorited some reports, in which case it is *Favorite*.

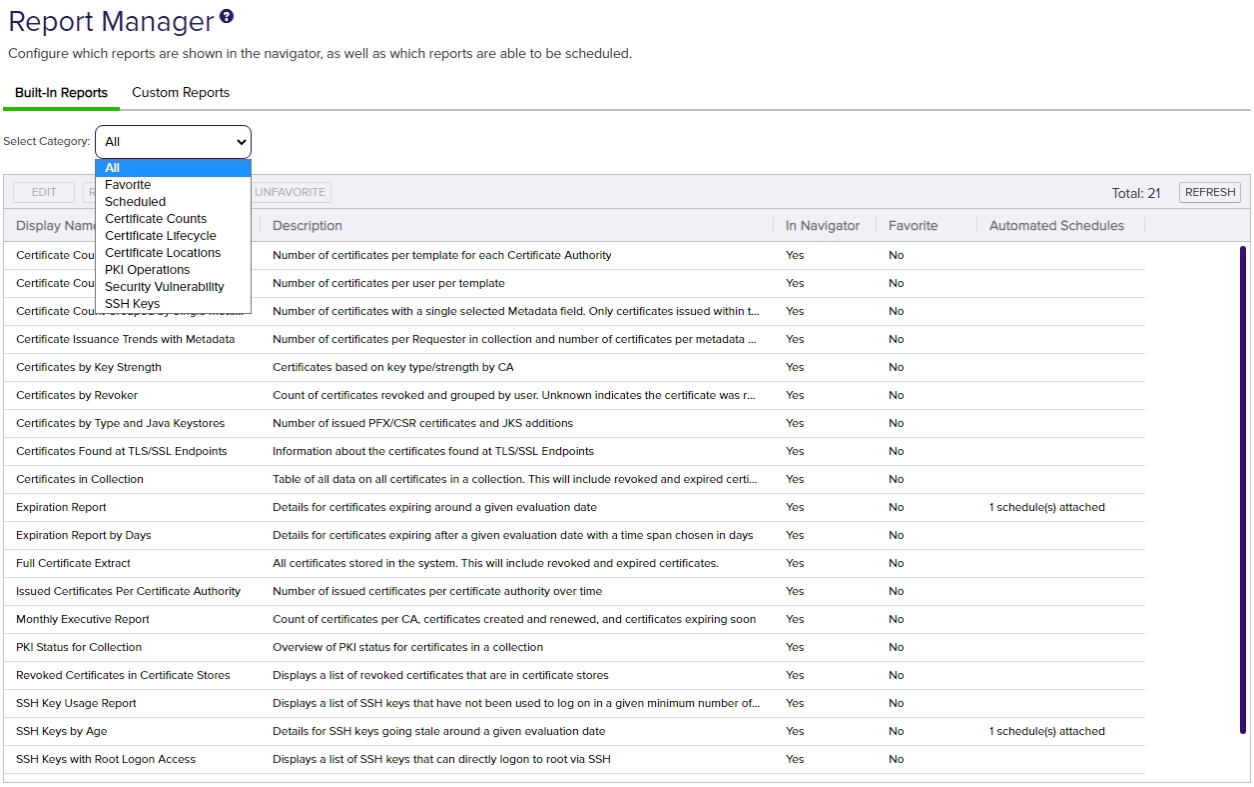


Figure 84: Report Manager Grid

Tip: If you **Save** a new certificate collection, or **Save** a change to an existing certificate collection, that change will be immediately reflected in the collection data used to display certificate collections on dashboards and reports. The data used by the dashboards and reports is stored in an intermediate table that is updated immediately. It will also continue to be updated periodically (approximately every 20 minutes by default as configured by the *Dashboard Collection Caching Interval* application setting) by the Keyfactor Command Service (see Application Settings: Console Tab on page 602).

Report Manager Operations

From the Report Manager you can run reports on demand, edit reports (modify how a report displays, change the parameter definitions and add or change the schedule(s) used to run the report), or delete reports. From the top grid menu you can also quickly change the *Favorite* setting for a report.

Run a Report

You can run a report on demand from the Report Manager page.

1. In the Management Portal, browse to *Reports > Report Manager*.
2. On the Report Manager page, highlight the report you wish to run in the grid and click **Run Report** from the top grid menu or the right click menu.
3. Populate the parameters as desired (see Parameters Tab on page 130 for more information on parameters).
4. Click **Generate**. The report will display immediately in the open window. The report can be exported to Excel, PDF or CSV, as available for that report, via the **Export** button at the end of the report.

Editing a Report and Scheduling a Report for Delivery

You can modify how a report displays, change the parameter definitions for running a report and add or change the schedule(s) used to deliver the report.

To edit an existing report:

1. In the Management Portal, browse to *Reports > Report Manager*.
2. On the Report Manager page, highlight the report you wish to modify in the grid and click **Edit** from the grid menu or the right click menu.
3. In the Report Manager dialog, edit the available options as needed.
4. Click **OK** to save the new or changed report details.

Details Tab

The most common edit to make on an existing report would be to check or uncheck the **Show in Navigator** box to add or remove the report from display on the Reports top menu, or to check or uncheck the **Favorites** box on the **Details** tab. The **Ignore Renewed Certificates** box will be available for reports that use collections to enable de-duplication (see the tip below). The **Uses Collection** box is for information only. It will be grayed out and checked for reports that use collections.

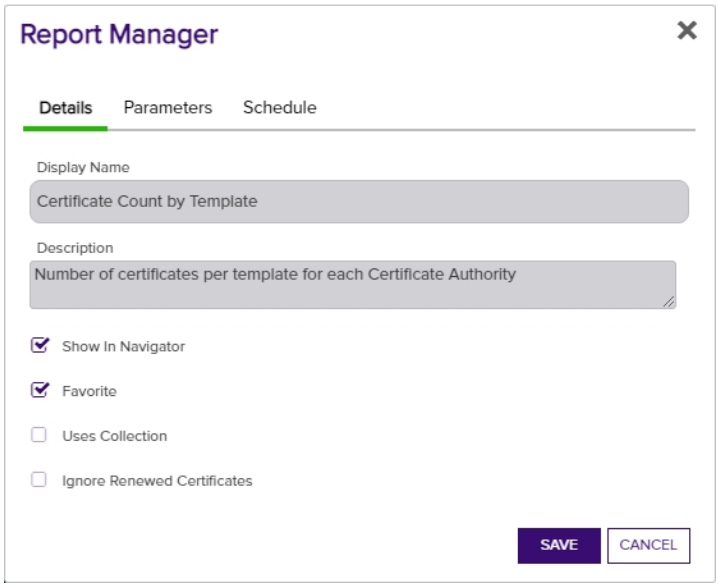


Figure 85: Edit a Report in Report Manager Details Tab

Tip: When de-duplication is enabled for a report, the report results will include only the most recently issued certificate if there is more than one certificate that matches the de-duplication criteria. De-duplication is enabled for a report by checking the *Ignore Renewed Certificates* box on the Details tab of the report configuration. De-duplication can only be enabled for reports that use certificate collections—the *Uses Collection* box on the Details tab. The *Uses Collection* setting is not user-configurable.
De-duping is configured on a certificate collection by setting the *Ignore renewed certificate results by* option when saving a certificate collection (see Saving Search Criteria as a Collection on page 42). Certificate collections may be configured to be de-duplicated based on the certificate common name, distinguished name, or principal name (or not at all). Only certificates that share all the EKUs (e.g. Client Authentication and Server Authentication) as well as the same CN, DN or UPN will be eliminated as duplicates. If a certificate has more than one EKU and at least one EKU does not match an otherwise similar certificate with matching CN, DN or UPN, it will not be eliminated.

For example, if the de-duplication logic was set to DN and the report would include these two certificates:

- Certificate one:
  - DN: CN=apps-srvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication
- Certificate two:
  - DN: CN=apps-srvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication

- Issued Date: December 1, 2020
- Expiration Date: January 1, 2022

- Issued Date: December 15, 2020
- Expiration Date: December 14, 2021

The de-duplication logic would be triggered because the DNs and EKUs match. The report would include certificate two and leave out certificate one. Notice that certificate two is retained even through certificate one expires after certificate two. This is because certificate two was issued after certificate one.

Now imagine that the de-duplication logic is set to CN and the report would include these two certificates:

- Certificate one:
  - DN: CN=appsrvr14.keyexample.com,OU=IT,O=Key Example, Inc.,L-L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication
  - Issued Date: December 1, 2020
  - Expiration Date: January 1, 2022
- Certificate two:
  - DN: CN=appsrvr14.keyexample.com,OU=HR,O=Key Example, Inc.,L-L=Chicago,ST=IL,C=US
  - EKUs: Server Authentication, Client Authentication
  - Issued Date: December 15, 2020
  - Expiration Date: December 14, 2021

Although the DNs for these certificates do not match, the CNs still do, so this matches the de-duplication logic of CN. However, the EKUs for these two certificates do not match, since only one of them includes Client Authentication. In this case, both certificates would appear on the report.

Parameters Tab

The Parameters tab will display all of the parameters for that specific report and allow you to configure default values to be used when the report is run from the Report Manager **Run Report** action button and what values default when adding a new schedule. You may also change the display name and description of the parameter.

To edit a parameter, select the **Parameters** tab, highlight the desired parameter in the parameters grid and click **Edit**, or double click the row. The *Parameters* dialog will open. Only those fields which can be edited will be enabled on the parameters details page. A change of the **Display Name** will change the name of parameter on the *Parameters* tab . A change of the **Description** will change the name of the description field on the *Schedule* tab. A change to the **Default Value** will define the value to use when the report is run from the Report Manager **Run Report** action button and what values default when adding a new schedule.

Tip: Some reports parameters use the **Add**/**Edit** button at the bottom of the dialog to open a Default Value dialog for to populate that parameter.

Note: The parameter fields will vary depending on the report selected. The parameters shown correspond to the specific parameters for each report. For more information on the parameters for a specific report, see the individual report under Reports on page 91.

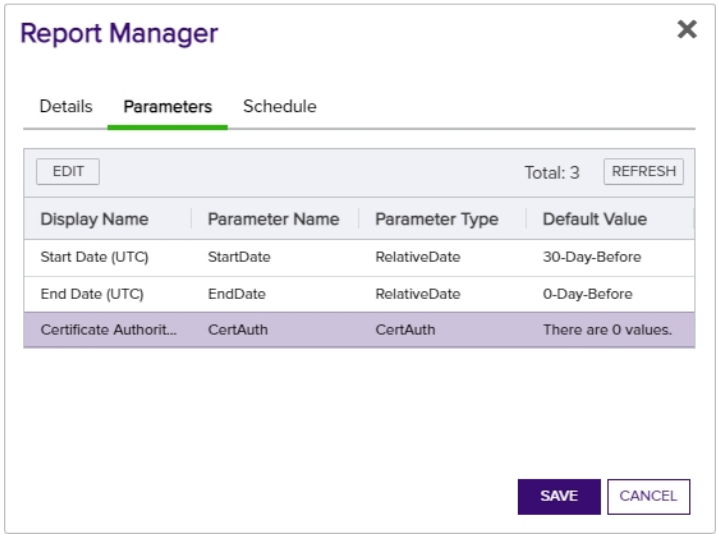


Figure 86: Edit a Report in Report Manager Parameters Tab

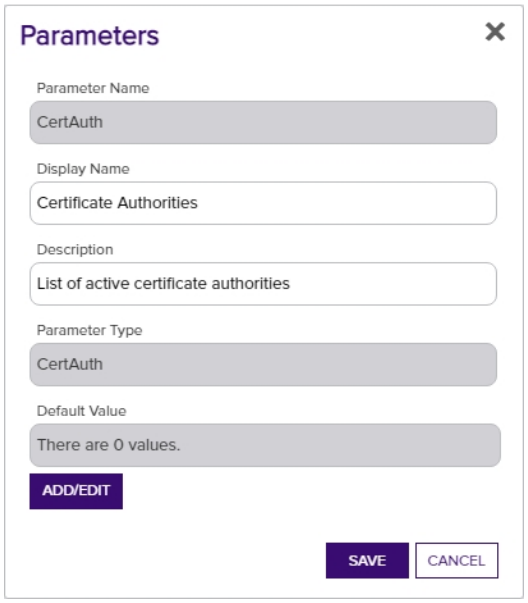


Figure 87: Report Manager Parameters Tab: Parameter Details

Schedule Tab

To add, edit, or delete a report delivery schedule, select the **Schedule** tab and choose the desired action. Any scheduled reports will appear on the schedule tab page. You can create multiple schedules with different parameters and recipients for the same report.

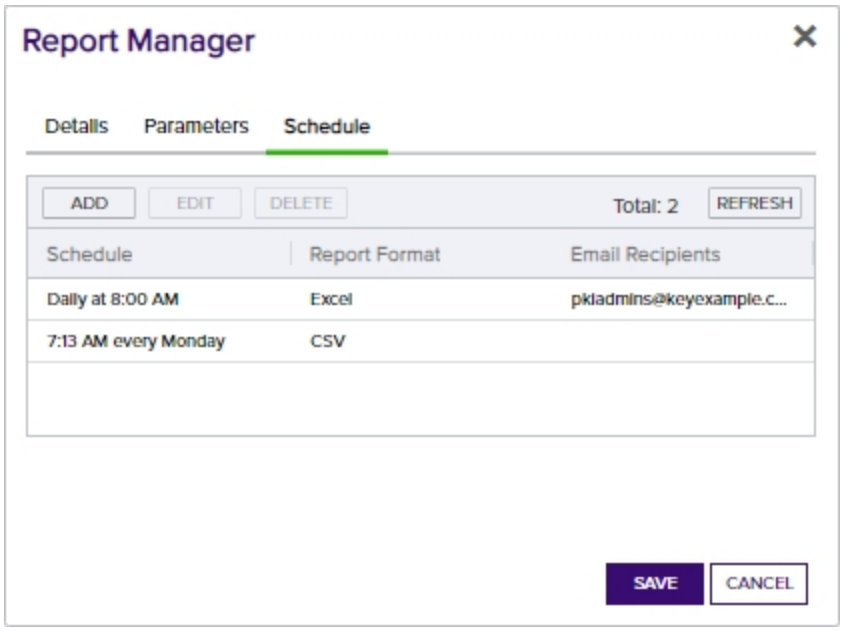


Figure 88: Edit a Report in Report Manager Schedule Tab

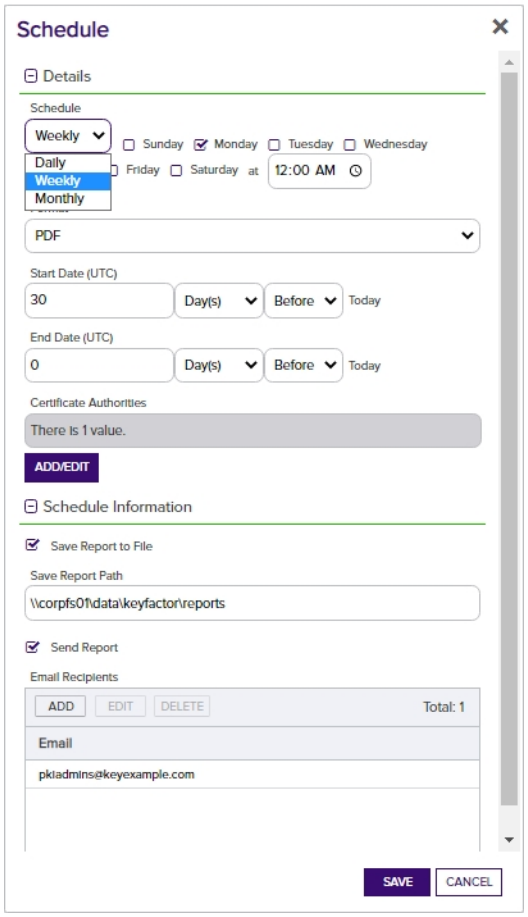


Figure 89: Edit a Report in Report Manager Schedule Tab - Add/Edit page

Note: Report scheduling is limited by collection permissions. Users in roles that have *Reports: Read and Modify* permissions will also need to have *Read* collection permissions on individual collections or global *Read* permissions for Certificates to have the ability to add, edit and delete schedules associated with collections. Any users without global *Read* permissions for Certificates will not have access to add, edit and delete schedules for any collections for which they do not have collection *Read* permissions in addition to *Reports* permissions.

Details section

- **Schedule**: Choose the schedule by selecting Daily, Weekly or Monthly from the dropdown, then choosing the day or date, and the time to run the report.

- **Report Format**: The available report formats are PDF, Excel and CSV*.

  Note: *The CSV format is only available on reports that contain all the data within a single section (such as the Certificates in Collection report) rather than broken out into multiple sections (such as the Expiration Report).

  Note: *CSV format is not available for custom reports with multiple tables.

- **Dynamic Parameters**:

  Depending on the parameters specific to the report, you will use either a entry field, a dropdown or click the **Add/Edit** to open the selection window for the report parameters for the specific schedule you are working on.

  - Some reports are based on a certificate collection, so one must be selected.
  - Some reports allow you to set an evaluation date for the report other than the current date so that you can, for example, run an Expiration Report time shifted to 1 month in the future to see what the expiration picture will look like in a month's time or compare last year to this year.
  - Some reports allow you to include custom metadata (see Certificate Metadata on page 710) in the report output.
  - Some reports allow you to select specific templates or CAs for reporting.

Schedule Information Section

- **Save Report to File**: You can choose to save your report to file by ticking the **Save Report to File** box, in which case you must provide a network path to which the file will be written in the **Save Report Path (relative to the server)** field. You will be given a warning message if the network path cannot be resolved. Although the record can still be saved with a path that doesn't resolve correctly, the report may fail to run if the path still does not resolve at the time the report runs.

  Note: The path for saved reports must be provided in UNC format (\\server-name\sharename\path) and must be accessible from the Keyfactor Command administration server. In addition:

  - Do not use a trailing "\" in the report path.
  - Ensure that the service account for the Keyfactor Command Service has permission to write to the location where you want the outputted report to be saved.

- When scheduling a report, schedule it for at least 10 minutes in advance of the current time if you wish it to run soon. If you want to run it faster than that, the Keyfactor Command Service will need to be restarted.

- **Send Report**: You can choose to deliver your report via email by ticking the **Send Report** box, in which case you must provide at least one recipient in the **Recipients** field at the bottom of the dialog.

Tip: For an explanation of the parameters specific to each report, see the section in the documentation for that specific report under Reports on page 91.

Important: Scheduled reports will not run if the Keyfactor Command Service is stopped.

Deleting a Report

To delete a report

1. In the Management Portal, browse to *Reports > Report Manager*.
2. On the Report Manager page, highlight the report you wish to delete in the grid and click **Delete** from the right-click menu.

Note: Only user-defined reports can be deleted. Built-in reports cannot be deleted. If you prefer not to see a built-in report, you may opt to remove the report from the menu by unchecking the **Show in Navigator** option.

2.1.5 Enrollment

The enrollment function in the Keyfactor Command Management Portal allows PKI administrators to request certificates by either submitting a certificate signing request (see CSR Enrollment on the next page) or by directly entering request information to receive a certificate delivered as a PFX file (see PFX Enrollment on page 146). The certificate file is available for immediate download via the browser or installation into a certificate store providing that the enrollment succeeds and the template used does not require manager approval. An option is also provided to generate a certificate signing request within Keyfactor Command. When you do this, the private key generated as part of the CSR generation process is stored—encrypted—in the Keyfactor Command database (see CSR Generation on page 142).

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

See Application Settings: Enrollment Tab for configuration settings that apply to the enrollment functions in the Keyfactor Command Management Portal. Some enrollment functions are also affected by template settings. See Configuring System-Wide Settings on page 382 and Configuring Template Options on page 387 for more information.

Note: The app pool service account must be set with permissions on the CA itself, in order to enroll via the CA in Keyfactor Command.

Important: Direct enrollment (without use of a Keyfactor CA gateway) is only supported for CAs in the forest in which Keyfactor Command is installed and any forests in a two-way trust with this forest. To do a cross-forest enrollment (with a forest in a two-way trust with the Keyfactor Command forest), Keyfactor Command requires that the root and intermediate CA certificates from the trusted forest are installed in the trusted root/intermediate stores in the Keyfactor Command server.

2.1.5.1 CSR Enrollment

The certificate signing request (CSR) enrollment page provides the ability to submit a CSR and download the resulting certificate.

Important: Before you can use the CSR enrollment function, you must configure at least one template for enrollment by checking the **CSR Enrollment** box under **Allowed Enrollment Types** in the certificate template details. See Configuring Template Options on page 387.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

To request a certificate via CSR:

1. Generate a CSR. This can be done within the target application (e.g. Microsoft IIS), by using a tool such as certutil or OpenSSL, or by using the Keyfactor Command CSR generation tool (see CSR Generation on page 142).
2. In the Management Portal, browse to *Enrollment > CSR Enrollment*.
3. Paste your CSR into the **CSR Content** text area, with or without the BEGIN REQUEST/END REQUEST delimiters.

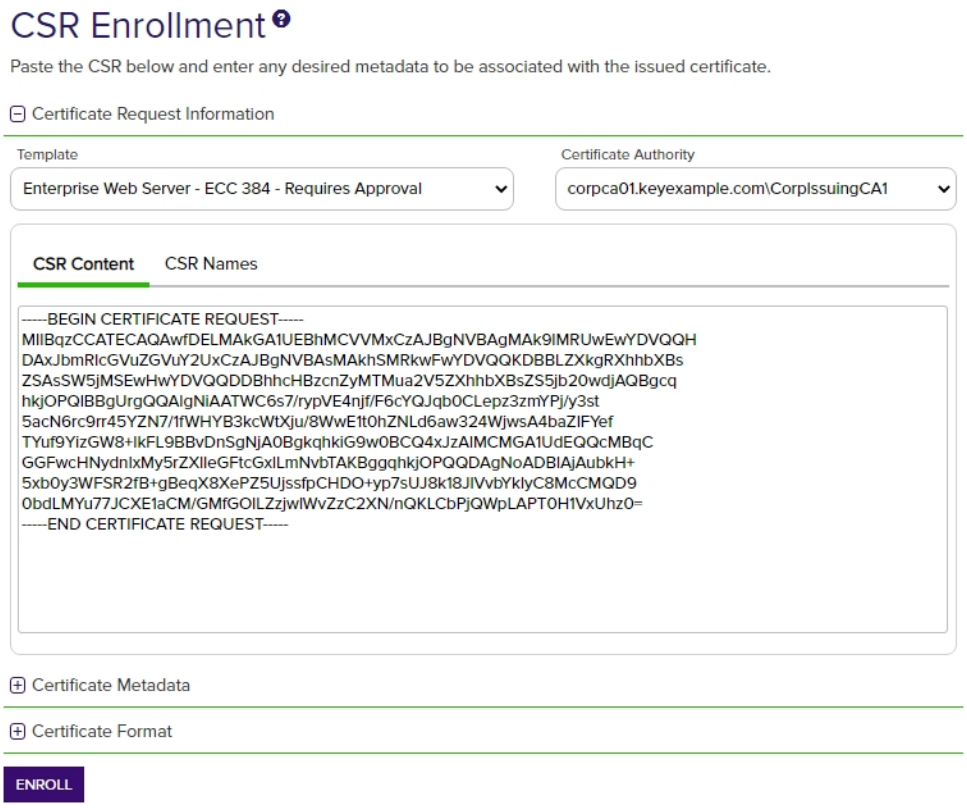


Figure 90: CSR Enrollment: CSR Content

4. The CSR contents will be parsed, and you will automatically be switched to the **CSR Names** view. Review the data to be sure it is as expected.

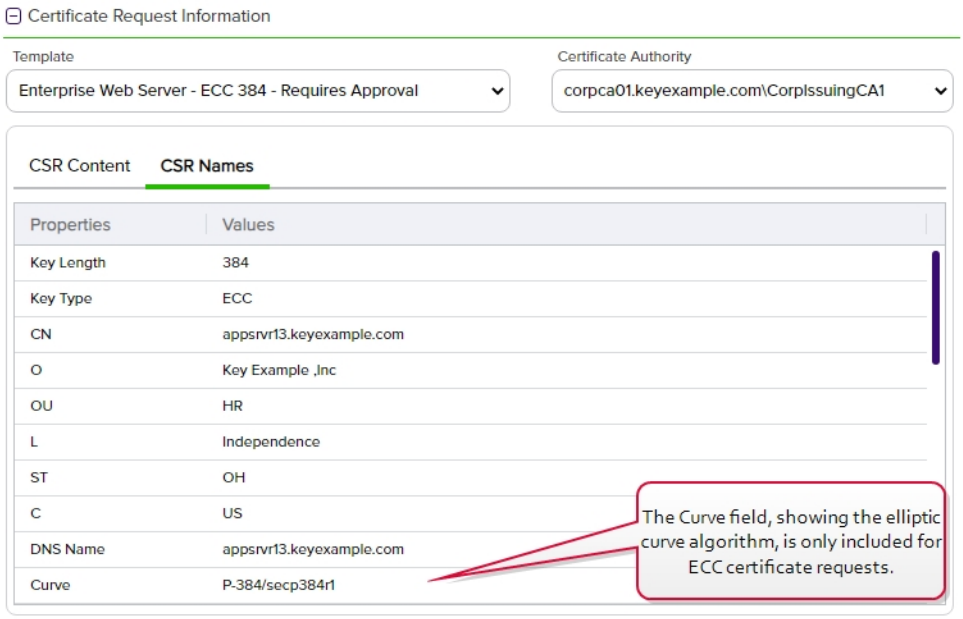


Figure 91: CSR Enrollment: CSR Names

Note: If a system-wide or template-level regular expression exists for a subject part or SAN, and the subject part or SAN is left blank, the regular expression will be applied to an empty string for that part. For example, if you have a regular expression on organization, but do not supply an organization, the regular expression will be applied to a blank string as if that were supplied as the organization.

5. If you are enrolling from an enterprise CA, select a certificate template from the **Template** drop-down. The templates are organized by configuration tenant (formerly known as forest). If you have multiple configuration tenants and templates with similar names, be sure to select the template in the correct configuration tenant.

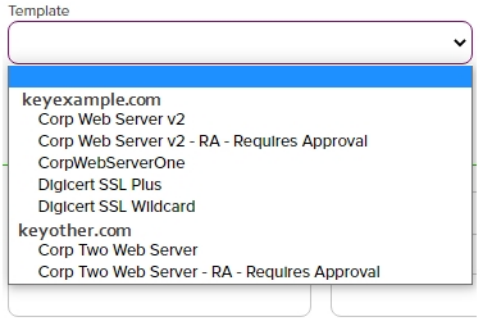


Figure 92: Select a Certificate Template

6. Select the **Certificate Authority** from which the certificate should be requested. Only CAs that have the selected template available for enrollment or are standalone, if you check the stand-alone CA box, will be shown.

Tip: If you are enrolling from a standalone CA, check the **Use a stand-alone CA** box instead of selecting a template. The check box for stand-alone CAs only appears if you have a stand-alone CA configured for enrollment.

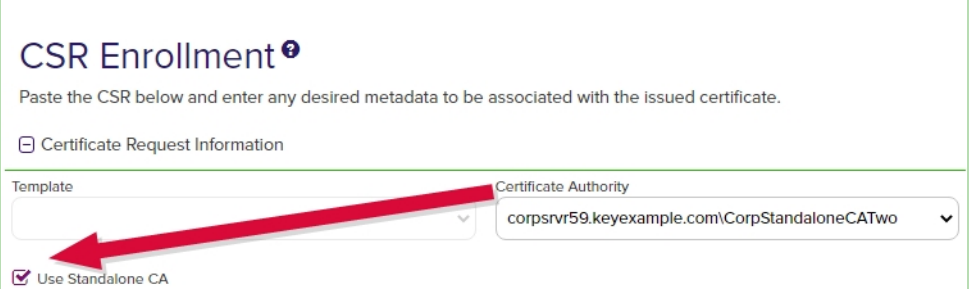


Figure 93: CSR Enrollment for Stand-Alone CA

7. The SAN section of the page appears if you enable the *Allow CSR SAN Entry* application setting (see Application Settings: Enrollment Tab on page 609). This option is disabled by default. In the Subject Alternative Names section of the page, click **Add** and select from the dropdown to enter one or more SANs for your CSR. Use the **Remove** action button to remove an existing SAN. The SAN field supports:

   - DNS name
   - IP version 4 address
   - IP version 6 address
   - User Prinicpal Name
   - Email

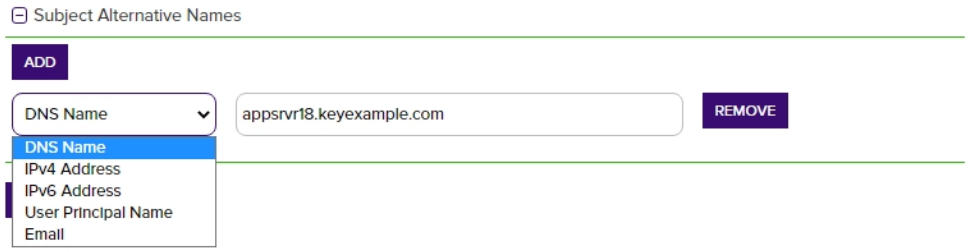


Figure 94: CSR Enrollment SAN options

Important: If the RFC 2818 compliance setting is enabled for the selected template (see Certificate Template Operations on page 381), your request must have at least one SAN either included in the original CSR or entered separately in this field, which matches the CN in the request.

Note: Entering SANs here may either append or overwrite the SANs in the CSR request depending on how the issuing CA is configured. Please be sure to check that the certificate has the correct SANs after issuance. Any SAN added automatically as a result of RFC 2818 compliance settings at the policy handler level will still be added alongside anything you add here. For more information, review the SAN Attribute Policy Handler for the Keyfactor CA Policy Module (see Installing the Keyfactor CA Policy Module Handlers on page 2846 in the *Keyfactor Command Server Installation Guide*).

8. If template-specific enrollment fields have been defined (see Enrollment Fields Tab on page 390) for the selected template, the fields will display in the Additional Enrollment Fields section. The types of fields shown could be either blank (string) fields or multiple choice drop-down fields depending on how they were configured on the template. **All additional enrollment fields are mandatory.**

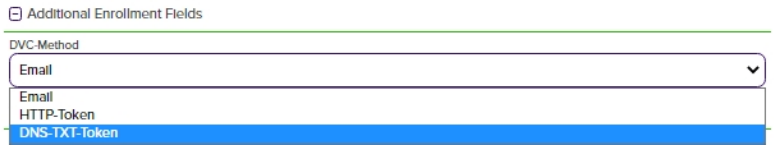


Figure 95: Populate Enrollment Fields

9. In the Certificate Metadata section of the page, populate any defined certificate metadata fields (see Certificate Metadata on page 710 and Metadata Tab on page 393) as appropriate for the template. These fields may be required or optional depending on your metadata configuration. Required fields will be marked with *Required next to the field label. Any completed values will be associated with the certificate once it has been synchronized with Keyfactor Command. The order in which the metadata fields appear can be changed (see Sorting Metadata Fields on page 715).

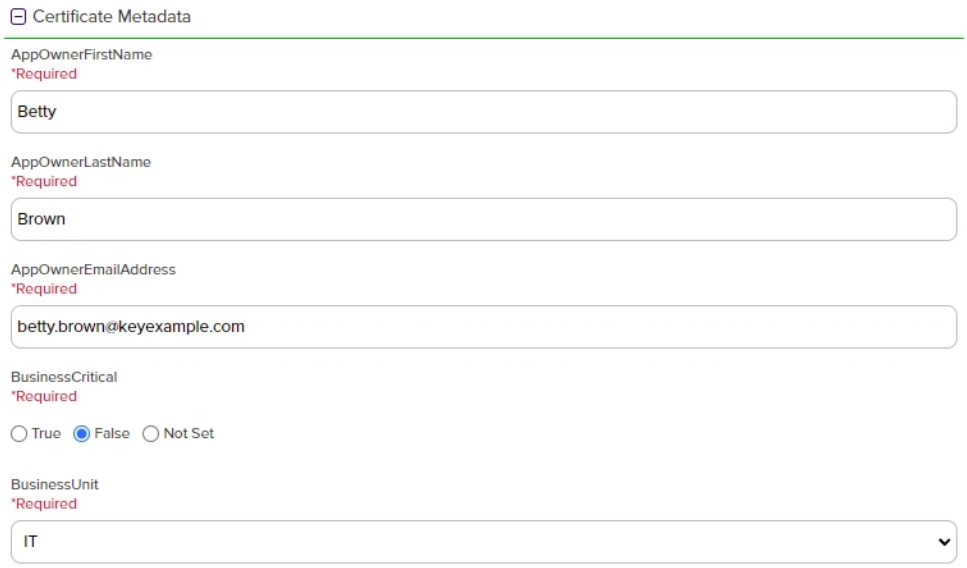


Figure 96: Populate Metadata Fields

10. At the bottom of the page, select the radio button for the desired encoding format (PEM or DER).

Figure 97: Select a Certificate Format

11. Click the **Enroll** button to begin the certificate request process.

    - If the request completes successfully, you'll see a success message and you'll be prompted by your browser to begin download of your certificate.
    - If the template you selected requires approval at the Keyfactor Command workflow level, you'll see a message that your request is suspended and is awaiting one or more approvals. The user(s) responsible for approving the request will be notified (if the workflow has been configured this way, see Adding, Copying or Modifying a Workflow Definition on page 237). You can use the My Workflows *Created by Me* tab (see Workflows Created by Me Operations on page 336) to check on the status of your request. If the Management Portal feature has been configured to send notification alerts when a certificate is issued following approval, you may receive an email message when your certificate is available for download. The email message may contain a download link. See Issued Certificate Request Alerts on page 188.

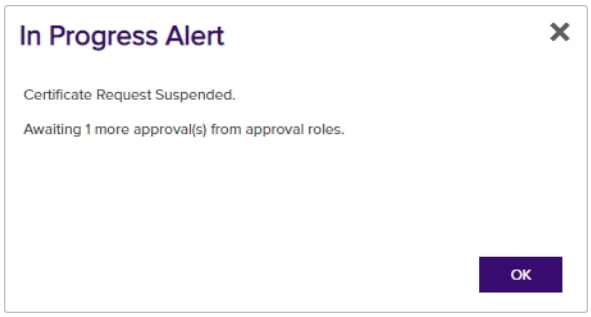


Figure 98: CSR Enrollment Completed Successfully—Awaiting Workflow Approval(s)

- If the template you selected requires manager approval at the CA level, you'll see a message that your request is pending. The user responsible for approving issuance of pending certificates will be notified (if that Management Portal feature is configured, see Pending Certificate Request Alerts on page 178). You can use the Certificate Requests page (see Certificate Requests on page 163) to check on the status of your pending request and complete the certificate download. If the Management Portal feature has been configured to send notification alerts when a pending certificate request is approved or denied, you may receive an email message when your certificate is available for download. The email message may contain a download link. See Issued Certificate Request Alerts on page 188 and Denied Certificate Request Alerts on page 197.

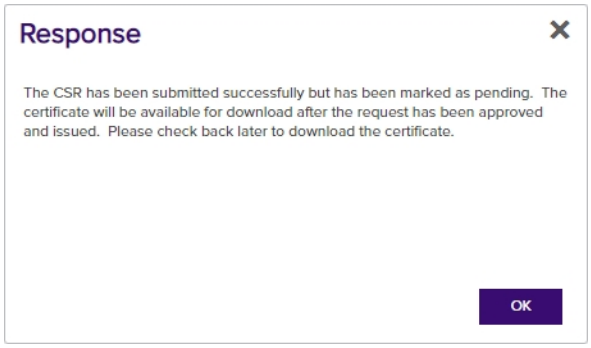


Figure 99: CSR Enrollment Completed Successfully—Pending Status

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

2.1.5.2 CSR Generation

The Certificate Signing Request (CSR) generation page provides the ability to enter a subject, SAN, key size, and template information and generate a CSR based on this information. You can then use this CSR to request a certificate using the CSR enrollment function (see CSR Enrollment on page 136) or any other enrollment method requiring a CSR.

When you use the CSR generation option, the encrypted private key of the request is stored in the Keyfactor Command database. When you generate a certificate using that CSR, it will be married together with the private key when the certificate synchronizes into the Keyfactor Command database. The certificate enrollment with the CSR does not need to be completed in Keyfactor Command (using CSR Enrollment) in order for the private key to be married with the certificate. Certificates enrolled outside of Keyfactor Command using CSRs generated within Keyfactor Command and synchronized via the CA synchronization process (see Certificate Authorities on page 349) or manually imported using the Add Certificate option (see Add Certificate on page 74) will also be married with their private keys.

To generate a CSR:

1. In the Keyfactor Command Management Portal, browse to *Enrollment > CSR Generation*.
2. In the Certificate Request Details section of the page:
    a. Select a **Template**, if desired. The templates are organized by configuration tenant (formerly known as forest). If you have multiple configuration tenants and templates with similar names, be sure to select the template in the correct configuration tenant.

    > **Important:** The template will not be included in the CSR. The template is referenced in order to retrieve key and other information to help populate the CSR. In addition, the CSR generation function supports template-level regular expressions for both subject parts and SANs. If system-wide and template-level regular expressions exists for the same field and you select a template, the template-level regular expression is applied.
    > If you choose to select a template during CSR generation, you will need to choose the same template during CSR Enrollment, because the CSR file will contain elements from the template which may conflict with other template configurations.

    b. Select a **Key Algorithm** and **Key Size** for your CSR. If you have selected a template, these dropdowns will be limited to the values supplied by the template. If the template supplies only one value for key algorithm and/or size, these dropdowns will be grayed out. When enrolling with the template, the key size of the request is validated against the template key size.

    > **Note:** The supported key algorithms for a certificate template are determined based on global template policy, individual template policy, and the template's supported algorithm.
    > When configuring template-level policies for key information, only key sizes that are valid for the algorithm will be available, according to the global template policy, the template policy, and the supported key sizes. For PFX and CSR generation, you will be offered the option to select the **Key Algorithm** and **Key Size** for the enrollment in dropdowns if the selected template with applied policy settings supports more than one of these. If, after applying Keyfactor Command policy to the returned template there is only one value for key algorithm and size, these dropdowns will be grayed

out. If for some reason an algorithm comes back as supported, but no key sizes are available, that algorithm should not appear. When selecting an ECC key size, the curve for that key size will be displayed.

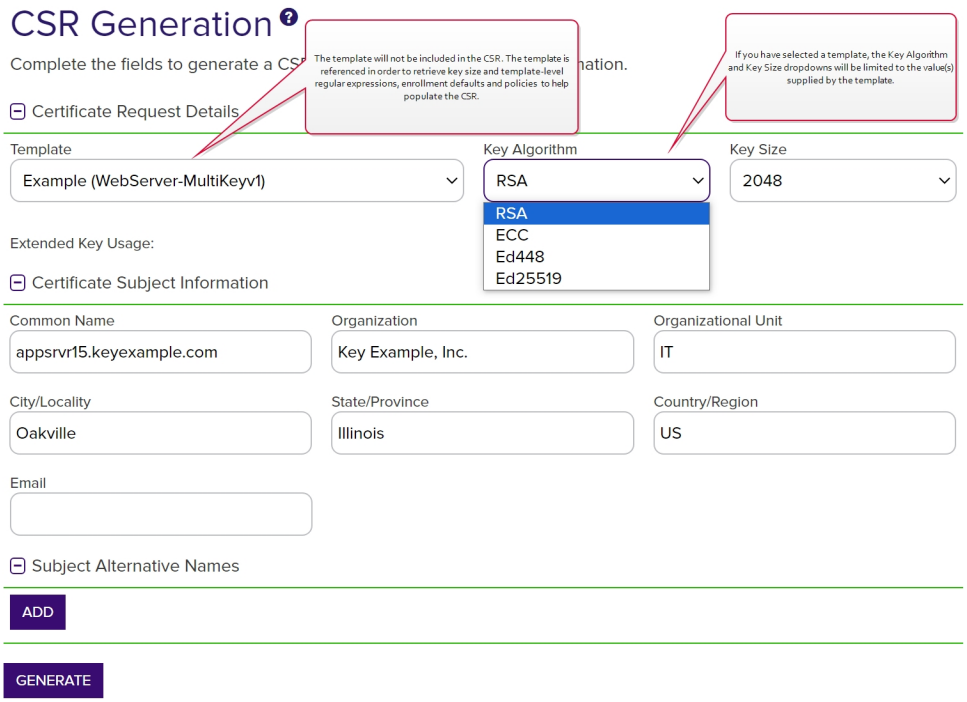


Figure 100: CSR Generation

3. In the Certificate Subject Information section of the page, enter appropriate subject information for your CSR.

   Note: Some subject fields may be automatically populated by system-wide or template-level enrollment defaults. You may override the system-populated data, if desired. Any system-wide or template-level regular expressions will be used to validate the data entered in the subject fields. System-wide or template-level policies will affect the request. For more information, see Certificate Template Operations on page 381. Subject data may also be overridden after an enrollment request is submitted either as part of a workflow (see Update Certificate Request Subject\SANs for Microsoft CAs on page 288) or using the *Subject Format* application setting (see Application Settings: Enrollment Tab on page 609).

4. In the Subject Alternative Names section of the page, click **Add** and select from the dropdown to enter one or more SANs for your CSR. Use the **Remove** action button to remove an existing SAN.

Important: If the template you selected has the RFC 2818 compliance setting enabled, the DNS name will be automatically populated with the Common Name (CN) and will be set to read only.

Note: If the CSR generated has multiple SANs, they will not be overridden by the template default settings, nor the RFC 2818 compliance settings.

The SAN field supports:

- DNS name
- P version 4 address
- IP version 6 address
- User Prinicpal Name
- Email

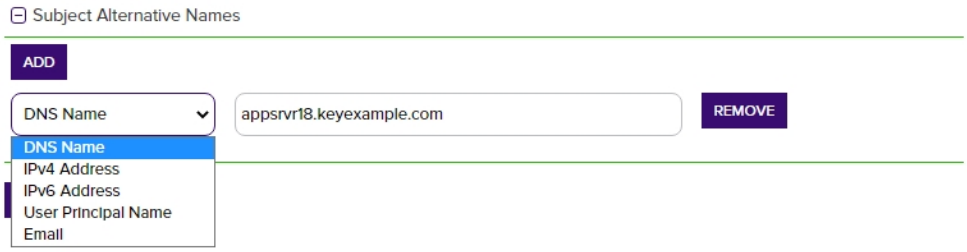


Figure 101: CSR Generation SAN Options

5. At the bottom of the page, click the **Generate** button. You will see a success message. If any template-level or system-wide regexes have been applied to any fields on the CSR and failed you will receive a notice at the top of the CSR generation page indicating the error as defined on the template (whether template or system-wide settings prevail).

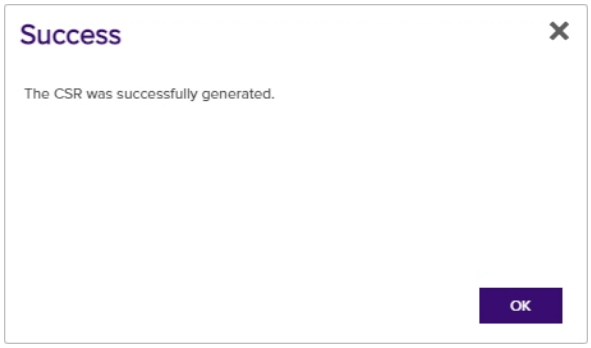


Figure 102: CSR Generation Success

6. Save or open your CSR once it has been successfully generated.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

2.1.5.3 Pending CSRs

The Pending CSRs page allows you to see if you have any outstanding certificate signing requests that have been generated but not used for enrollment. From here, you can download them so that you can use them for enrollment or delete them if they are no longer needed. To download, highlight the selected row, right-click and choose **Download** from the right-click menu, or choose the **Download** action button at the top of the grid.

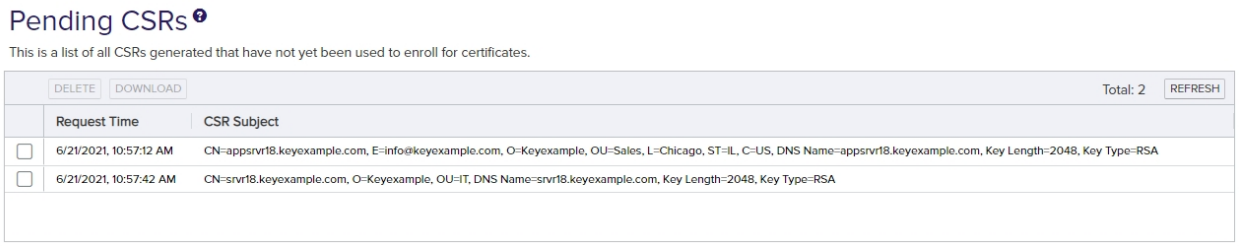


Figure 103: Pending CSRs

The pending certificate grid includes these fields:

- Request Time
  The date and time the CSR request was submitted in Keyfactor Command.
- Subject Name
  The subject name of the CSR, including key size, key type, and SANs, if applicable.

The CSRs can be sorted by clicking on the **Request Time** column header in the results grid. Click the column header again to reverse the sort order. The results grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may also be adjusted by click-holding and dragging the line separating two column headers.

2.1.5.4 PFX Enrollment

The PFX Enrollment page provides the ability to submit a certificate request and download the resulting PFX certificate file. Given the power involved in allowing a user to generate his or her own subject name and automatically receive a certificate in this subject name, Keyfactor recommends that permissions for this feature are only given to very trusted users and/or that you consider making use of Keyfactor Command workflow with a RequireApproval step (see Adding, Copying or Modifying a Workflow Definition on page 237).

Important: Before you can use the PFX enrollment function, you must configure at least one template for enrollment by checking the **PFX Enrollment** box under **Allowed Enrollment Types** in the certificate template details. In addition, if you wish to use a template that requires *CA certificate manager approval*, you must enable one of the **Private Key Retention** options in the certificate template details. See Certificate Template Operations on page 381.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

You can expand and collapse sections of the PFX enrollment page by clicking on the plus/minus icon to the left of each section title.

To request a certificate via PFX:

1. In the Keyfactor Command Management Portall, browse to *Enrollment > PFX Enrollment*.
2. If you are enrolling from an enterprise CA, select a certificate template from the **Template** drop-down. The templates are organized by configuration tenant (formerly known as forest). If you have multiple configuration tenants and templates with similar names, be sure to select the template in the correct configuration tenant. If you are enrolling from a standalone CA, check the **Use a stand-alone CA** box instead of selecting a template.

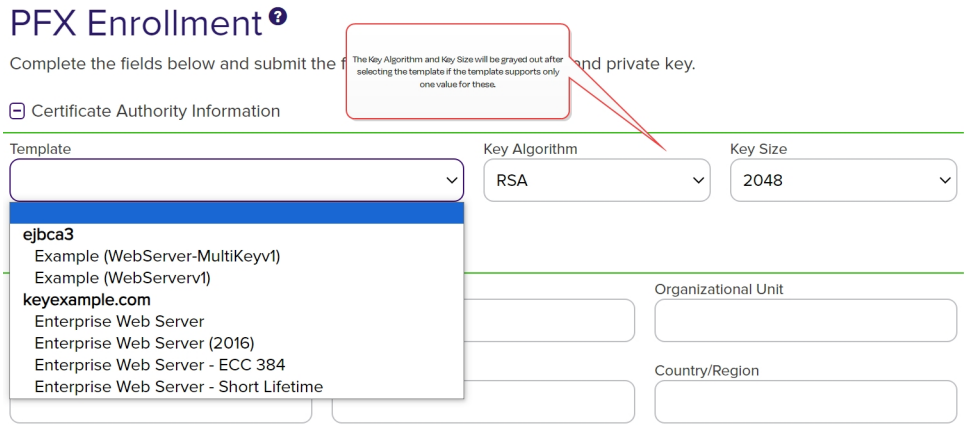


Figure 104: Select a Certificate Template

Note: The supported key algorithms for a certificate template are determined based on global template policy, individual template policy, and the template's supported algorithm.

When configuring template-level policies for key information, only key sizes that are valid for the algorithm will be available, according to the global template policy, the template policy, and the supported key sizes. For PFX and CSR generation, you will be offered the

option to select the **Key Algorithm** and **Key Size** for the enrollment in dropdowns if the selected template with applied policy settings supports more than one of these. If, after applying Keyfactor Command policy to the returned template there is only one value for key algorithm and size, these dropdowns will be grayed out. If for some reason an algorithm comes back as supported, but no key sizes are available, that algorithm should not appear. When selecting an ECC key size, the curve for that key size will be displayed.

Tip: The check box for stand-alone CAs only appears if you have a stand-alone CA configured for enrollment.

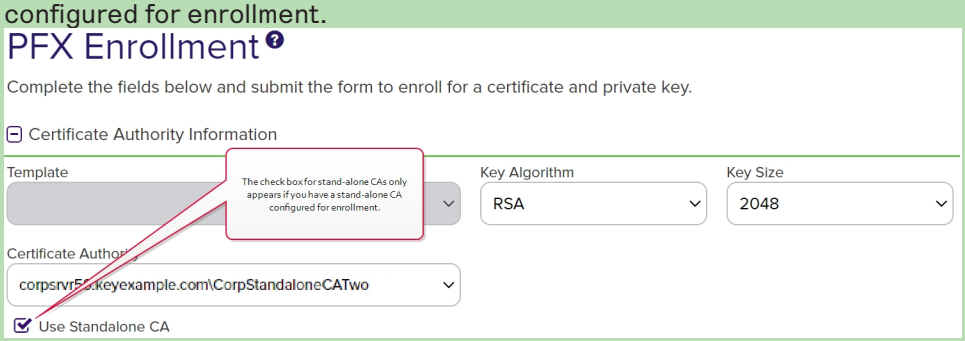


Figure 105: PFX Enrollment for Stand-Alone CA

Tip: If you select an ECC template, the elliptic curve algorithm for the template appears below the Template dropdown.

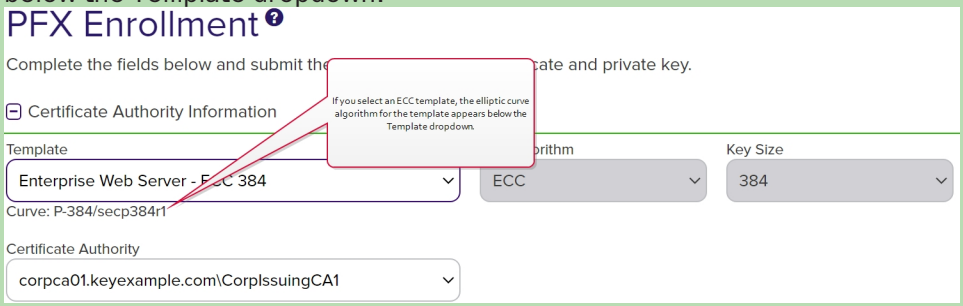


Figure 106: PFX Enrollment for ECC Template Displaying Elliptic Curve

3. Select the **Certificate Authority** from which the certificate should be requested. Only CAs that have the selected template available for enrollment or are standalone, if you check the stand-alone CA box, will be shown.

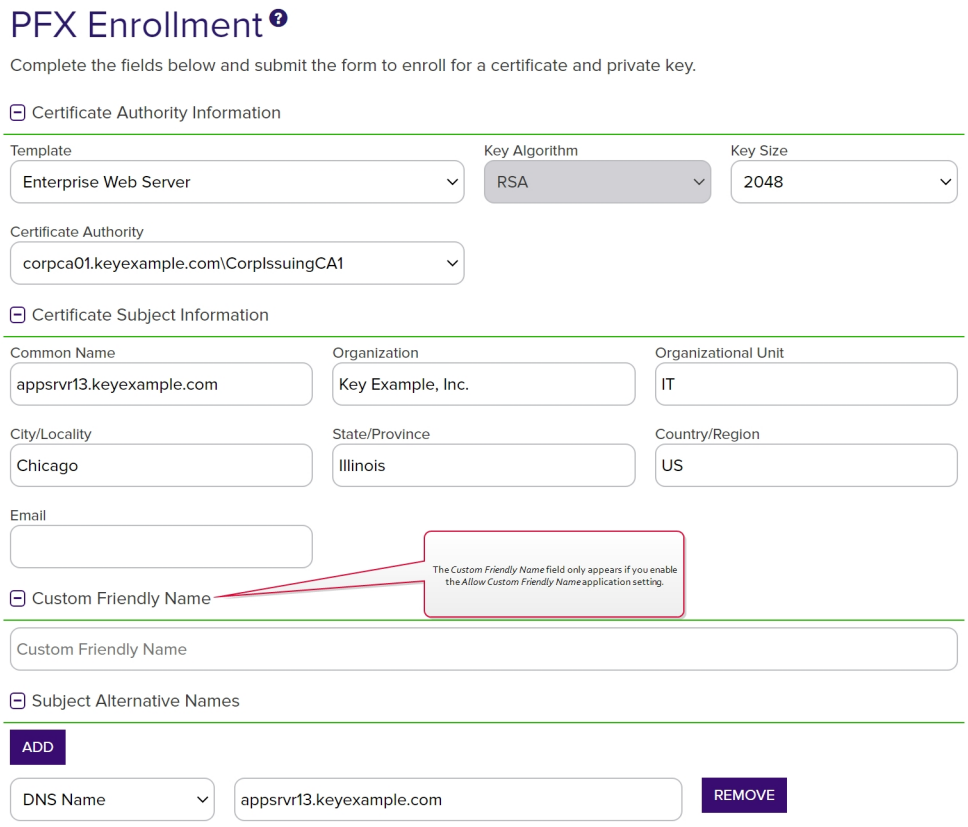


Figure 107: PFX Enrollment

> **Note:** If a system-wide or template-level regular expression exists for a subject part or SAN, and the subject part or SAN is left blank, the regular expression will be applied to an empty string for that part. For example, if you have a regular expression on organization, but do not supply an organization, the regular expression will be applied to a blank string as if that were supplied as the organization

4. In the Certificate Subject Information section of the page, populate the fields as appropriate for the certificate being requested. Although Keyfactor Command does not strictly require the **Common Name**, the product does ship with a default regular expression requiring a value for this field since it is typical for a CA to require this unless the template is set to populate the subject from Active Directory. This regular expression may have been altered in your environment (see the below note).

> **Note:** Some subject fields may be automatically populated by system-wide or template-level enrollment defaults. You may override the system-populated data, if desired. Any system-wide or template-level regular expressions will be used to validate the data entered in the subject fields. System-wide or template-level policies will affect the request. For more information, see Certificate Template Operations on page 381. Subject

data may also be overridden after an enrollment request is submitted either as part of a workflow (see Update Certificate Request Subject\SANs for Microsoft CAs on page 288) or using the *Subject Format* application setting (see Application Settings: Enrollment Tab on page 609).

5. If enabled, add a friendly name in the Custom Friendly Name section of the page. This section only appears if the *Allow Custom Friendly Name* application setting is set to *True*. If the *Require Custom Friendly Name* application is set to *True*, a value is required in this field. For more information, see Application Settings: Enrollment Tab on page 609.

6. In the Subject Alternative Names (SANs) section of the page, add SANs if needed. If the RFC 2818 compliance option has been enabled for the template (see Certificate Template Operations on page 381), the first SAN field will automatically populate with a DNS SAN matching the CN when you enter the CN be set to *Read Only*. Click the **Add** button to add SAN fields.

   Important: If the template you selected has the RFC 2818 compliance setting enabled, the DNS name will be automatically populated with the Common Name (CN) and will be set to read only.

   The SAN field supports:

   - DNS name
   - IP version 4 address
   - IP version 6 address
   - User Prinicpal Name
   - Email

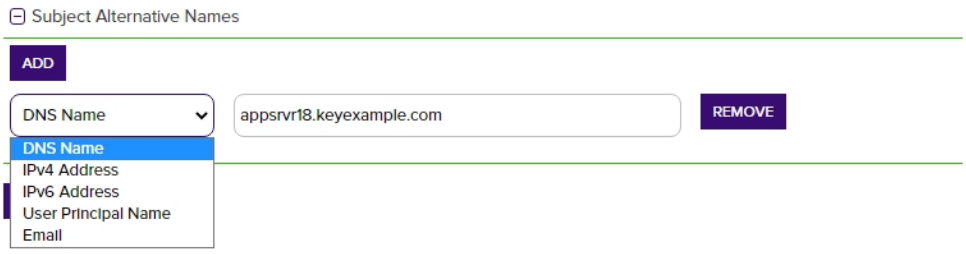


Figure 108: PFX Enrollment: SAN Options

   This field is not required unless the RFC 2818 compliance option has been configured in the template policy.

7. If template-specific enrollment fields have been defined (see Enrollment Fields Tab on page 390) for the selected template, the fields will display in the Additional Enrollment Fields section. Additional enrollment fields have a data type of either string or multiple choice. String fields will appear as a text box; Multiple choice fields will appear as a dropdown. All additional enrollment fields are required.

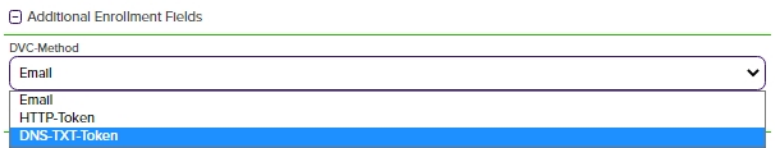


Figure 109: Populate Enrollment Fields

8. In the Certificate Metadata section of the page, populate any defined certificate metadata fields (see Certificate Metadata on page 710 and Certificate Template Operations on page 381) as appropriate for the template. These fields may be required or optional depending on your metadata configuration. Required fields will be marked with *Required next to the field label. Any completed values will be associated with the certificate once it has been imported into Keyfactor Command. The order in which the metadata fields appear can be changed (see Sorting Metadata Fields on page 715).

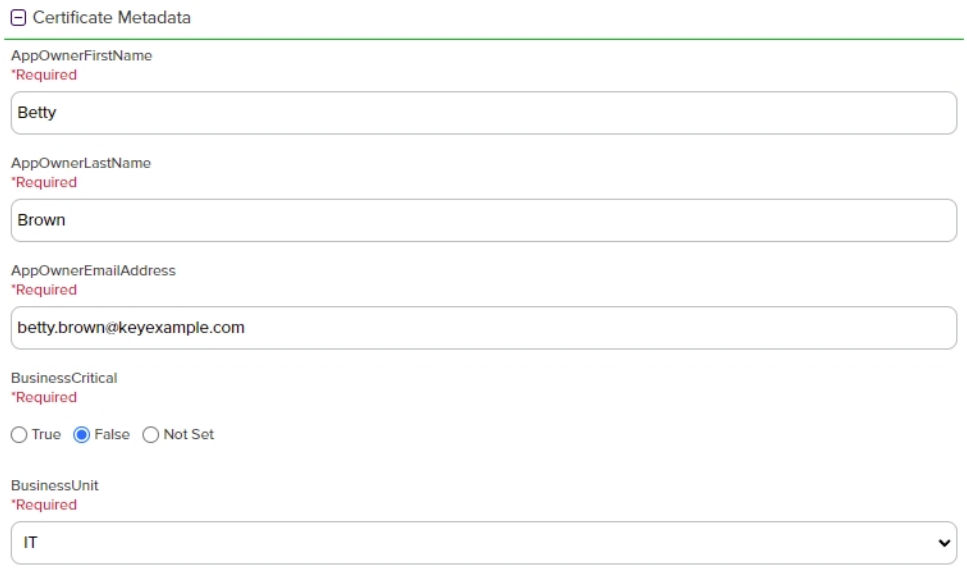


Figure 110: Populate Metadata Fields

9. If enabled, in the Password section of the page, check the **Use Custom Password** box and enter and confirm a custom password to use in securing the PFX file. This section only appears if the *Allow Custom Password* application setting is set to *True*. The value in the *Password Length* field in application settings is shown for guidance when entering a password. For more information about both of these settings, see Application Settings: Enrollment Tab on page 609.

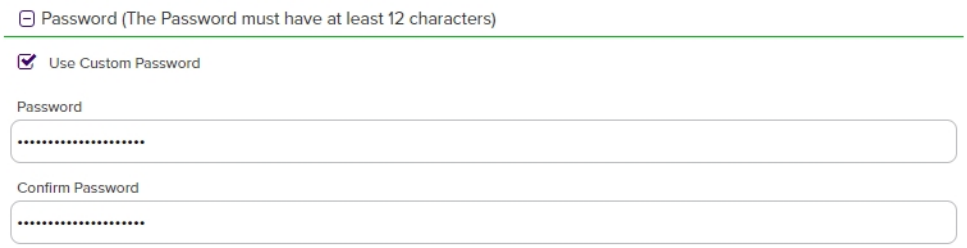


Figure 111: Set a Custom Password

10. In the Certificate Delivery Format section of the page, select either **Direct Download**—to download the certificate immediately—or **Install into Certificate Stores**—to schedule the certificate to be added to a configured certificate store. If you do not have any configured certificate stores, the Install into Certificate Stores option will not appear.

Direct Download

If you selected Direct Download, choose whether to include the certificate chain with the returned certificate by toggling **Include Chain**. If you toggle Include Chain to on, choose a **Chain Order** and specify either **End Entity First** or **Root First** order. The option to specify the order will only be available if the selected format supports it, otherwise the order will always be **End Entity First**.

The **Use Legacy Encryption** option applies only to certificates downloaded in PFX format. If this option is toggled to **On**, the PFX file is encrypted using the historical algorithm (3DES/SHA1/RC2). If this option is toggled to **Off**, the newer algorithm set provided by Windows (AES256/SHA256/AES256) is used instead. The default for the PFX enrollment page can be set with the *Use Legacy Encryption* application setting (see Application Settings: Enrollment Tab on page 609). The toggle defaults to off.

The supported **Formats** for output are:

- PFX (PKCS#12)
- PEM (one PEM file, with optional chain)
- ZIP PEM (A ZIP file with individual PEM files for certificate, private key, and each part of the chain, if included)
- JKS

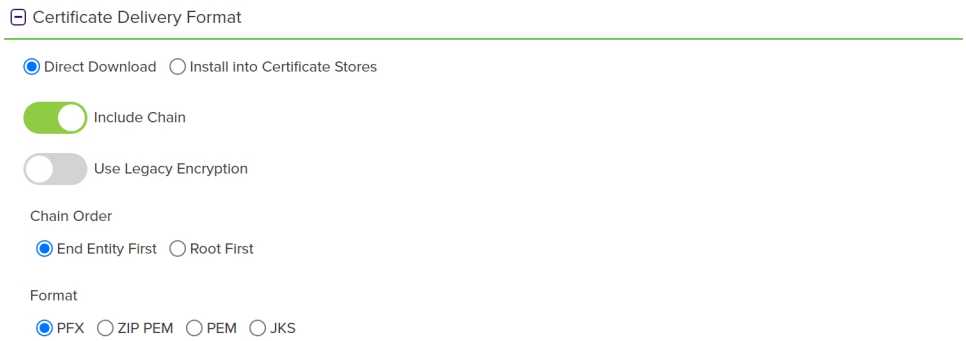


Figure 112: Delivery Format PFX Enrollment

Install into Certificate Stores

If you have any certificate stores defined, you may opt to install the certificate directly into one or more certificate stores on enrollment. If you choose to do this, the certificate will not be available for download on this page.

To install a certificate into a certificate store, select the **Install into Certificate Stores** radio button and then click the **Include Certificate Stores** button. This will cause the *Select Certificate Store Locations* dialog to appear. Make your certificate store selections in this dialog as described in *Select Certificate Store Locations*, below, and click **Include and Close**. You will then see some additional fields on the enrollment page. Populate these as per *Add to Certificate Stores* and *Information Required for Certificate Stores*, below.

Select Certificate Store Locations

The *Select Certificate Store Locations* dialog allows you to run queries against your certificate store list to select which store(s) to deploy a selected certificate to. **Check** the box next to each certificate store location to which you want to distribute the certificate.

Note: Only compatible certificate stores and only stores in containers to which you have permissions are shown on the grid.

Tip: You may change the search results by using the search fields at the top of the dialog. All of the Keyfactor Command grid search features are available to assist your search. See Using the Certificate Store Search Feature on page 410 for more information on the available search fields. The default search criteria is *AgentAvailable is equal to True*.

The actions on the *Select Certificate Store Locations* dialog are:

- **Include**

  Click this to add the selected certificate store(s) to your certificate selection and leave the search dialog open for further searches.

- **Include and Close**

  Click this to close the search dialog and add the selected certificate store(s) to your certificate selection, which will then be displayed and ready for updates as per the instructions in *Add to Certificate Stores*.

- **Close**

  Click this to cancel the operation and return to the main page with no certificate stores selected.

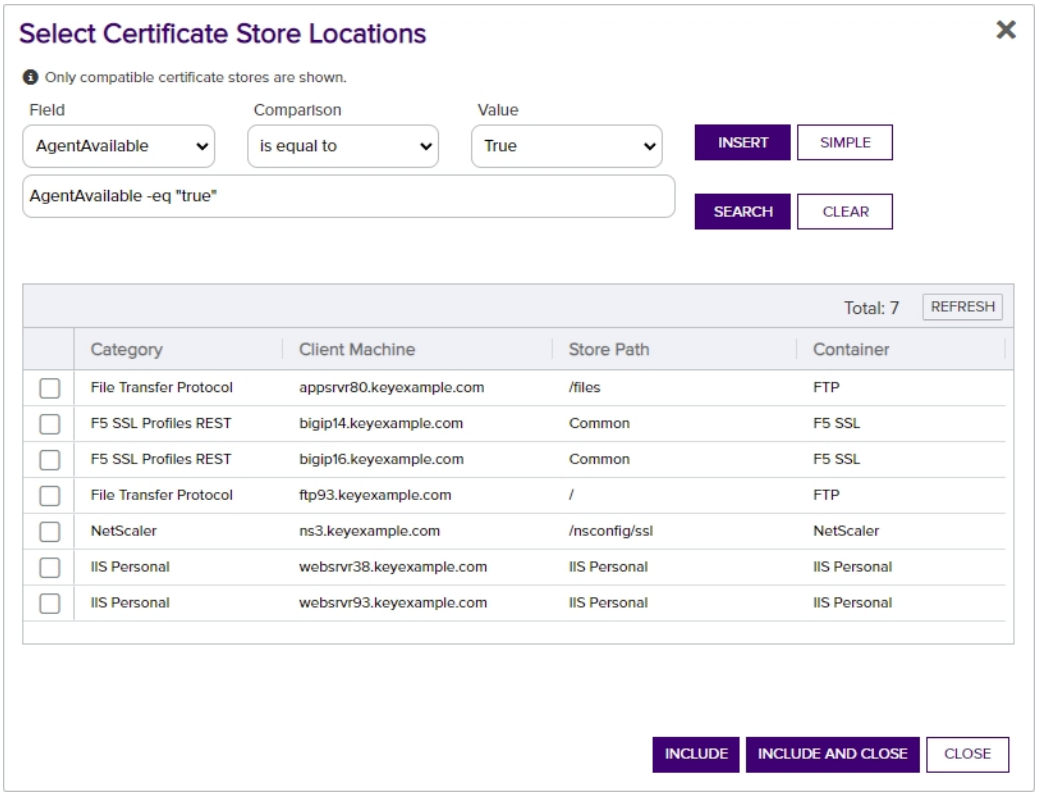


Figure 113: Select Certificate Store Locations Dialog

Add to Certificate Stores

The additional fields that appear on the page once you select at least one certificate store to distribute your certificate to include a grid section with a series of tabs that displays a tab for each type of certificate store selected with a list of the selected stores under each tab.

Above this section are global options that apply to the add job as a whole:

- **Schedule when to run the job for the certificate store**

  In the **Schedule** dropdown, select a time at which the job to add the certificate to the stores should run. The choices are *Immediate* or *Exactly Once* at a specified date and

time. If you choose *Exactly Once*, enter the date and time for the job. A job scheduled for *Immediate* running will run within a few minutes of saving the operation. The default is *Immediate*.

- **Include Certificate Stores**

  Open the *Select Certificate Store Locations* dialog again.

For each selected certificate store you can apply the following actions:

- **Overwrite**

  Check **Overwrite** below the grid to allow the selected certificate to overwrite any existing certificate in the same location with the same name or alias.

- **Alias**

  Add an **Alias** below the grid, if applicable, for the certificate store type. See the **Information Required by Certificate Store** section, below, for more information.

  Note: The tab heading of the certificate location will display an alert if an alias is required for the location.

- **Remove**

  Click **Remove** at the top of the grid to remove the selected certificate store from the page. The certificate will not be added to the store.

You may return to the *Select Certificate Store Locations* dialog by clicking **Include Certificate Stores** above the grid. The current selections will be retained.

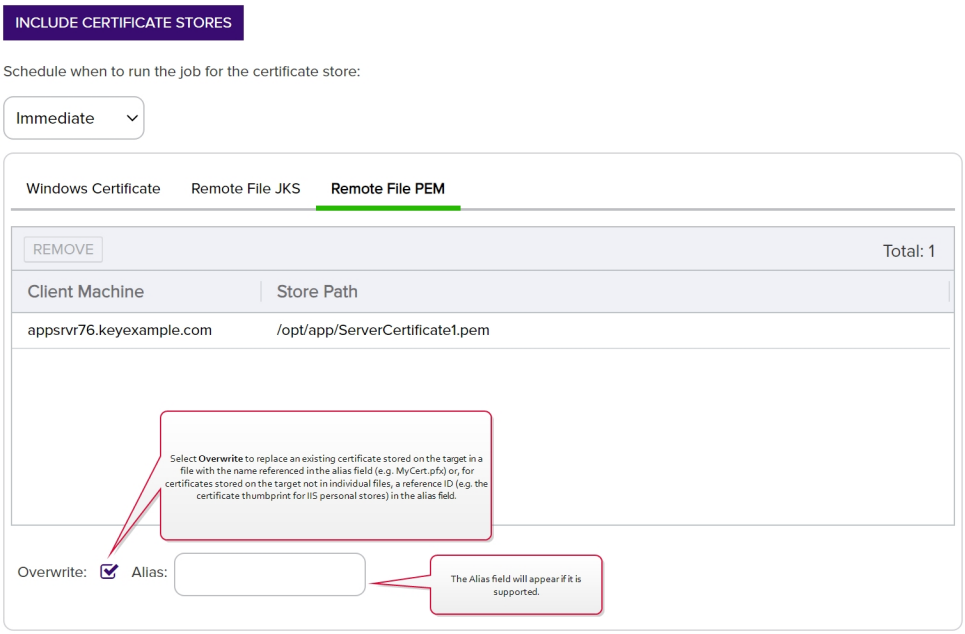


Figure 114: PFX Enrollment: Certificate Delivery Format

Figure 115: Alias Required System Alert on Enrolling

Information Required by Certificate Stores

Each type of certificate store has different requirements for providing an alias or other additional information.

The certificate store type fields that are relevant to certificate store use are:

- **Supports Entry Password**

  If your certificate store type has this enabled, you will have the option to enter a password for the certificate entry in the certificate store on the addition of an entry into the certificate store.

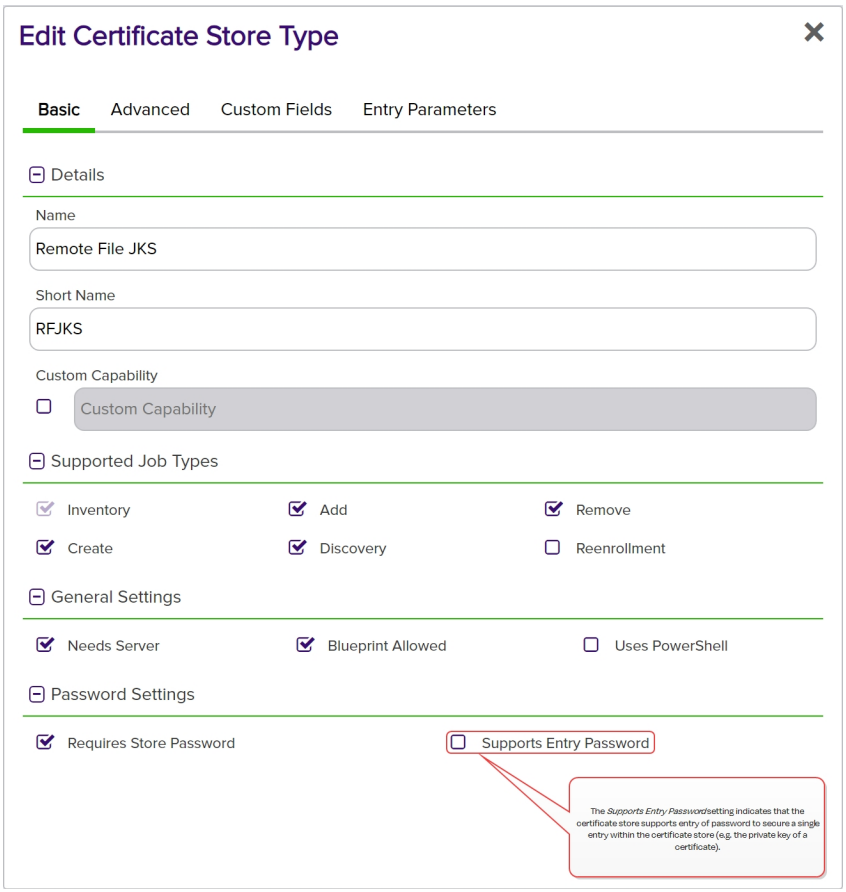


Figure 116: Certificate Store Type Configuration: Basic Tab

- **Supports Custom Alias**

  A value of *Required* indicates that a custom alias will be required when a certificate is added to a certificate store. *Optional* indicates an alias can be associated with the entry if desired. If your certificate store type sets this to *Forbidden*, the *Alias* field will not display when adding a certificate to a certificate store unless *Overwrite* is checked on the add page. In this case, you're not associating an alias with the certificate you're adding to the store but rather specifying the alias of the certificate already in the store that you wish to replace (in function) with the new certificate you're adding.

  The format of custom alias values varies depending on the certificate store type. In many cases, the alias is the thumbprint of the certificate. In some cases, it's the file name of the certificate file or a custom alias provided at the time the certificate was added to the certificate store. For instance:

- For an Amazon Web Services (AWS) store, the alias is the internal ID assigned by Amazon (the Amazon resource number or ARN). Provide the entire contents of the *Alias/IP* from this field when entering an alias for overwrite. For example:

  arn:aws:acm:us-west-2:220531701668:certificate/88e5dcfb-a70b-4636-a8ab-e85e8ad88780

- For F5 stores using the Keyfactor custom-built F5 Certificate Store Manager extension (see Installing Custom-Built Extensions on page 2940), the alias is the file name used to store the file in the device file system, minus the extension (e.g. use alias MyFile for a file named MyFile.crt). Aliases should be entered without spaces. Note that certificate names are case sensitive.

Note: Keyfactor Command will automatically strip out any spaces between the octets in thumbprints in the alias field, so it does not matter whether you enter the thumbprint with or without spaces.

Tip: When adding a certificate to a certificate store, you have the option to overwrite an existing certificate with the current certificate. If you choose this option, you will need to provide the alias of the certificate you wish to overwrite. Find the alias values by navigating to *Management Portal > Certificates > Certificate Search*. Select the certificate you wish to overwrite and double-click, or click **Edit**, from the grid header or right-click menu. Choose the **Locations** tab and double-click on the Location Type (this must have a number other than zero in the *Count* column) to open the details dialog. The *Alias* field holds the information that may be required for an overwrite.

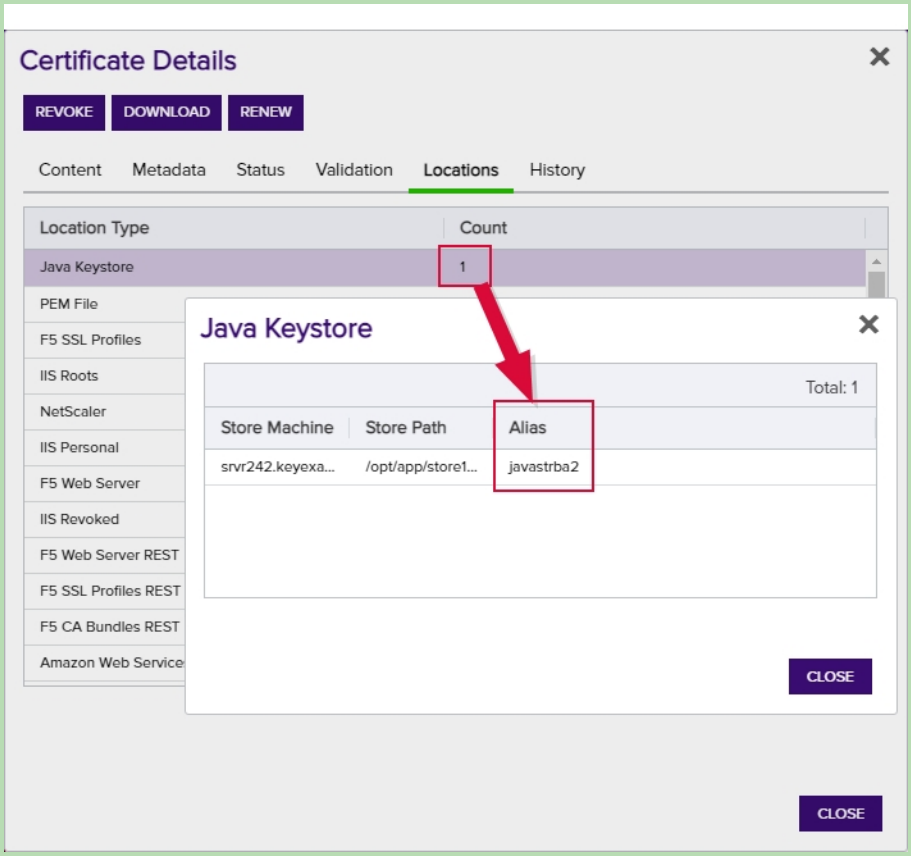


Figure 117: Example: Certificate Location Details for a JKS Location

- **Private Key Handling**

  When adding a certificate to a certificate store, if you select a certificate that does not have an associated private key, certificate stores with this option set to *Required* will not appear as available stores to which the certificate can be added. If this option is set to *Forbidden* and the selected certificate has a private key, the private key will be ignored and only the public key will be delivered to the target.

Note: Private keys are always available in PFX Enrollment.

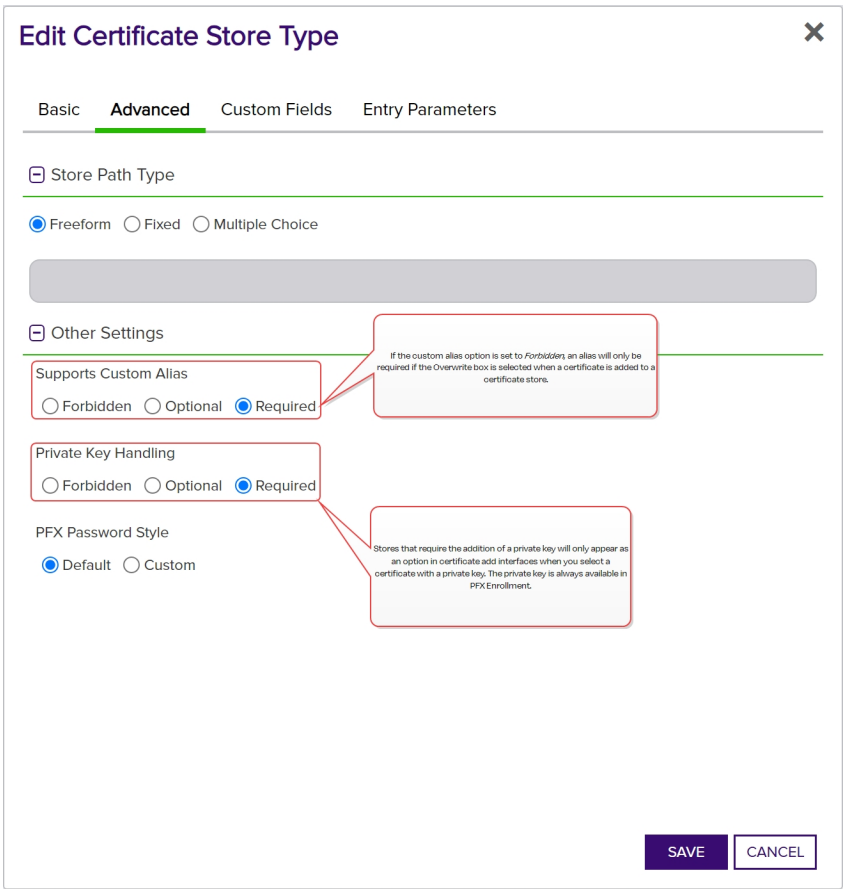


Figure 118: Certificate Store Type Configuration: Advanced Tab

- **Entry Parameters**

  Not all certificate store types will have entry parameters. The ones shown in Figure 119: Certificate Store Type Configuration: Entry Parameters Tab are for the custom *Windows Certificate* type for the Keyfactor custom-built IIS Certificate Store Manager extension (see Installing Custom-Built Extensions on page 2940).

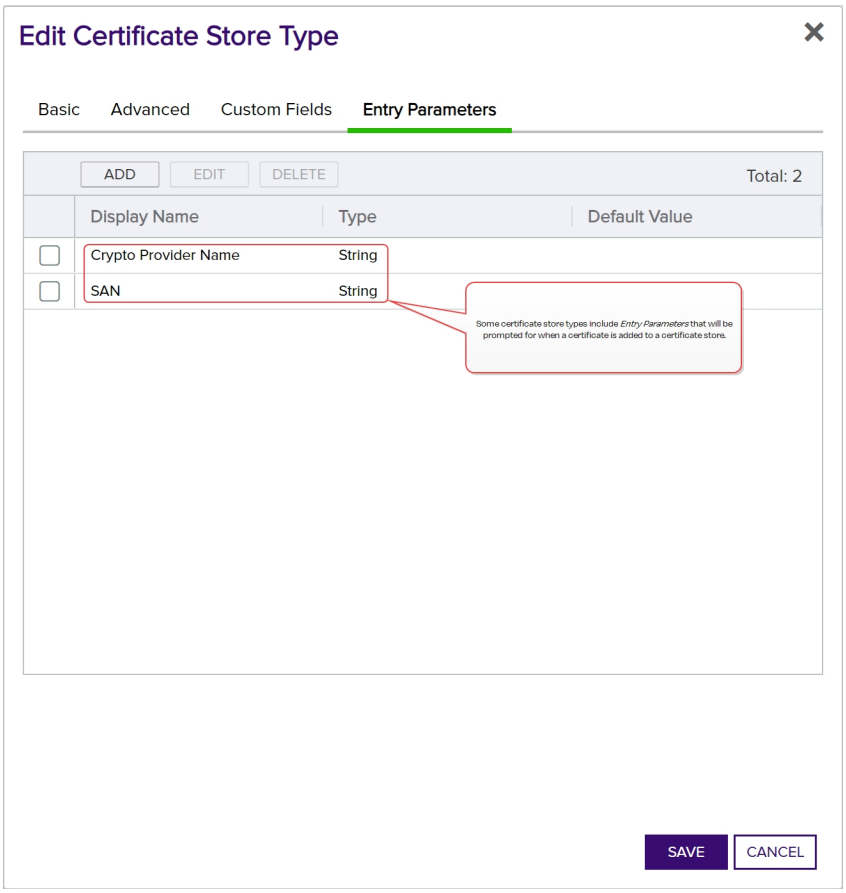


Figure 119: Certificate Store Type Configuration: Entry Parameters Tab

11. At the bottom of the page, click **Enroll** to begin the certificate request process.

    - If the request completes successfully, you'll see a success message. When the certificate is enrolled and issued the message will state the download format type, if the private key was included in the downloaded certificate, if the certificate chain was included in the downloaded certificate. If private key was not retained for the downloaded certificate, no password will be displayed a message will state that the private key and certificate can be obtained from the certificate search page.
    - You'll be prompted by your browser to begin download of your certificate unless you chose to install it directly into a certificate store. If you've configured PFX enrollment to use Windows authentication (the default) and have not selected the option to enter a custom password, you'll see a one-time password that has been generated to secure the PFX file. You will need this password in order to open the PFX file.
    - If you've configured the Keyfactor Command Management Portal to use basic authentication and you've configured the *Use Active Directory Password* application setting option to True, the message will indicate that the PFX file can be opened using the Active Directory domain password of the user making the request. For more information about configuring basic authentication versus Windows authentication, see Application Settings:

Enrollment Tab on page 609.

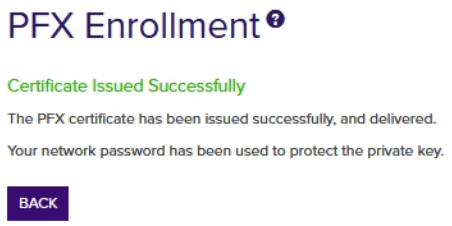


Figure 120: PFX Enrollment Completed Successfully—Network Password Used

Note: This option does not work when you authenticate to the Management Portal using Kerberos because Keyfactor Command does not have access to your credentials to apply your password to the PFX file.

- If the template you selected requires approval at the Keyfactor Command workflow level, you'll see a message that your request is suspended and is awaiting one or more approvals. The user(s) responsible for approving the request will be notified (if the workflow has been configured this way, see Adding, Copying or Modifying a Workflow Definition on page 237). You can use the My Workflows *Created by Me* tab (see Workflows Created by Me Operations on page 336) to check on the status of your request. If the Management Portal feature has been configured to send notification alerts when a certificate is issued following approval, you may receive an email message when your certificate is available for download. The email message may contain a download link. See Issued Certificate Request Alerts on page 188.

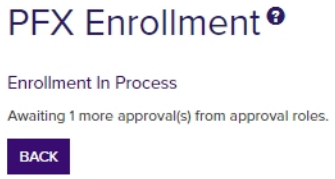


Figure 121: PFX Enrollment Completed Successfully—Awaiting Workflow Approval(s)

- If the template you selected requires manager approval at the CA level, you'll see a message that your request is pending. The user responsible for approving issuance of pending certificates will be notified (if that Management Portal feature is configured, see Pending Certificate Request Alerts on page 178). You can visit the Certificate Requests page (see Certificate Requests on the next page) to check on the status of your pending request and certificate search (see Certificate Search and Collections on page 19) to complete the certificate download. If the Management Portal feature has been configured to send notification alerts when a pending certificate request is approved or denied, you may receive an email message when your certificate is available for download. The email message may contain a download link. See Issued Certificate Request Alerts on page 188 and Denied Certificate Request Alerts on page 197.

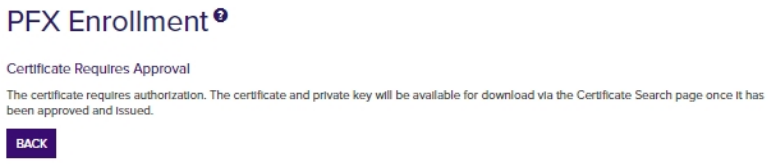


Figure 122: PFX Enrollment Completed Successfully—Pending Status

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

2.1.5.5 Certificate Requests

The Certificate Requests page shows certificate requests made to certificate authorities that have been configured to synchronize to the Keyfactor Command database and which have a status of pending, external validation or denied/failed. You can approve or deny pending certificates from this page (see Approving or Denying a Pending Certificate Request on page 165).

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Viewing Certificate Requests

The Certificate Requests grid has three tabs: **Pending**, **External Validation** and **Denied/Failed**. Select the appropriate tab to the view desired certificate requests. You may also filter the list shown by entering all or part of a **Requester Name** and clicking **Filter** to change which requests are displayed.

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230) do not appear on this page.

- **Pending**

  Typically a request in this state has been made using a template that requires manager approval at the CA level before issuance. The request may be approved or denied from this tab of the certificate requests page or though action on a *Pending Request Alert* (see Pending Certificate Request Alerts on page 178). When the pending requests tab is selected, you will see **Approve** and **Deny** buttons activated at the top of the grid. By clicking **Details**, you can view the

certificate details and **Approve** or **Deny** the request from the Certificate Request Details dialog. See Approving or Denying a Pending Certificate Request on the next page for more information.

- **External Validation**

  Certificate requests in this state require approval outside of Keyfactor Command. Certificates appearing on this tab generally are for requests made through one of the Keyfactor Command CA gateways using an EV certificate type. The requests appear here for reference only and cannot be approved or denied. Once a request has been approved using the cloud provider's EV approval process, the Keyfactor Command CA gateway and Keyfactor Command will import the issued certificate on the next synchronization. The synced certificate will move to the Certificate Search grid (see Certificate Search and Collections on page 19) and can be viewed there.

- **Denied/Failed**

  The denied/failed view shows requests that have been denied through Keyfactor Command as an action on the certificate requests page **Pending** tab though action on a *Pending Request Alert* (see Pending Certificate Request Alerts on page 178), or through a *POST /Workflow/Certificates/Deny* API request (see POST Workflow Certificates Deny on page 2484 in the *Keyfactor API Reference Guide*), but does not include requests denied directly from the CA outside of Keyfactor Command.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Monitoring > Alerts > Read

The certificate requests grid includes these fields:

Keyfactor Request ID
The reference ID of the request from the Keyfactor database.

Common Name
The requested common name of the request.

Distinguished Name
The requested distinguished name of the request.

Submission Date
The date on which the request was submitted.

Certificate Authority
The CA against which the request was made.

Template
The short name of the template used to make the request.

Requester
The user or entity that made the request.

State
The request status—failed, pending or external validation as per the tab selected.

By default, the grid sorts in descending order with the most recent certs at the top. The grid can be sorted in ascending or descending Submission Date order by clicking on the column header. he grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may also be adjusted by click-holding and dragging the line separating two column headers.

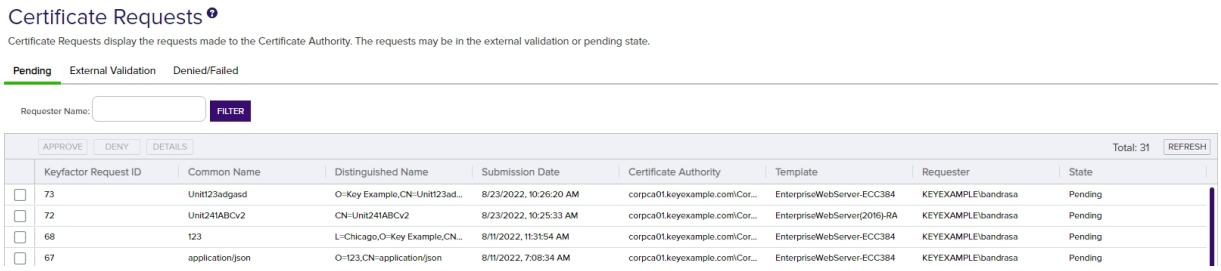


Figure 123: Certificate Requests Grid

The **Details** button appears activated for all views. The details page includes the SANs, metadata, and certificate stores scheduled for distribution for the request, in addition to the information shown on the main grid.

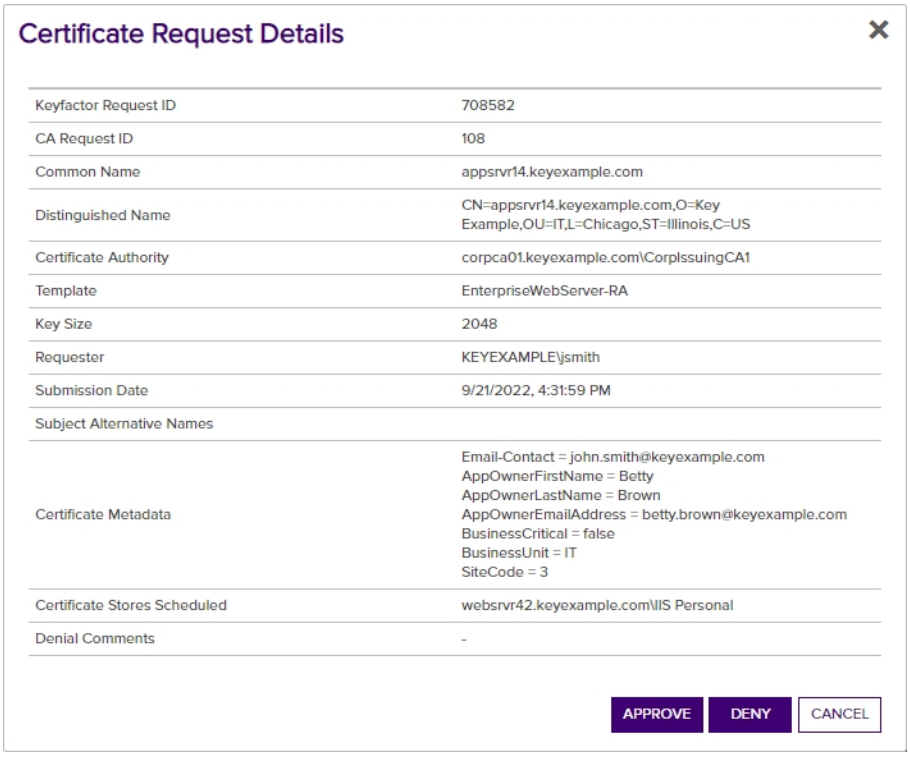


Figure 124: Certificate Request Details

Approving or Denying a Pending Certificate Request

On the **Pending** tab of the certificates requests grid you can view the **Details** of a certificate request that required manager approval at the CA level and choose to **Approve** or **Deny** it by clicking the action buttons at the top of the grid. You can also **Approve** or **Deny** the request from the Certificate Request Details dialog. The approve and deny operations can be done on multiple requests at once. To select multiple rows, click the checkbox for each row on which you would like to perform an oper-

ation, then select an operation from the top of the grid. The right-click menu only supports operations on one request at a time.

- When you deny a request, you will be prompted to enter a comment regarding the denial. These comments can be delivered to the requester or other interested party using a denied request alert (see Denied Certificate Request Alerts on page 197). When a certificate is denied, its status will change to failed and it will move from the pending grid tab to the denied/failed grid tab. The denial comments will display in the Certificate Request Details dialogue.
- When a request is approved on this page, the certificate will move to the Certificate Search grid (see Certificate Search and Collections on page 19) and can be viewed there. If you have configured issued certificate alerts (see Issued Certificate Request Alerts on page 188), the requester or other interested party will be notified immediately on approval.

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230) do not appear on this page.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Monitoring > Alerts > Read
Certificates > Requests Manage

Certificate requests with a pending status have generally either been requested using certificate templates requiring manager approval at the CA level or from a CA configured to send all requests to pending automatically.

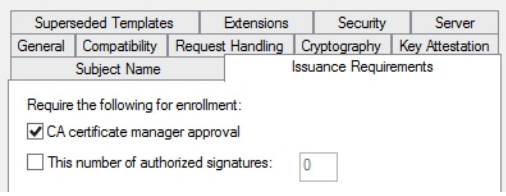


Figure 125: Certificate Template Requiring Manager Approval

Note: Certificate requests that require approval at the CA level are supported only for Microsort CAs and select CA gateways. This feature is not supported for EJBCA CAs. Use workflow for configuring Keyfactor Command-level approvals for EJBCA CAs (see Workflow Definitions on page 230).

2.1.6 Alerts

The options available in the Alerts section of the Management Portal are:

- Expiration Alerts below
  Create email notifications that alert administrators and/or end users when certificates are coming up for expiration.
- Pending Certificate Request Alerts on page 178
  Create email notifications that alert PKI administrators when a new pending certificate request is made.
- Issued Certificate Request Alerts on page 188
  Create email notifications that alert a certificate requester when a certificate he or she requested has been issued.
- Denied Certificate Request Alerts on page 197
  Create email notifications that alert a certificate requester when a certificate he or she requested has been denied.
- Key Rotation Alerts on page 203
  Create email notifications that alert end users and PKI administrators when an SSH key is nearing the end of its lifetime.
- Revocation Monitoring on page 210
  Define locations where certificate revocation lists (CRLs) and online certificate status protocol (OCSP) locations may be found and enable expiration notification alerts for them.

2.1.6.1 Expiration Alerts

Expiration alerts are used to send email notifications to certificate owners, users and/or administrators when a certificate is nearing or at expiration. The alerts can be customized to provide detailed information about the certificates along with, for example, instructions to end users on how to enroll for a replacement certificate.

Expiration Alert Operations

Expiration alerts are based on certificate collections. Before you can work with expiration alerts, you need to have created a certificate collection on which to base the alert (see Certificate Search and Collections on page 19).

Adding or Modifying an Expiration Alert

1. In the Management Portal, browse to *Alerts* > *Expiration*.
2. On the Expiration Alerts page, click **Add** from the top menu to create a new alert, or **Edit** from either the top or right click menu, to modify an existing one.
3. In the Certificate Expiration Alert Settings dialog, select your **Certificate Collection** in the first dropdown.

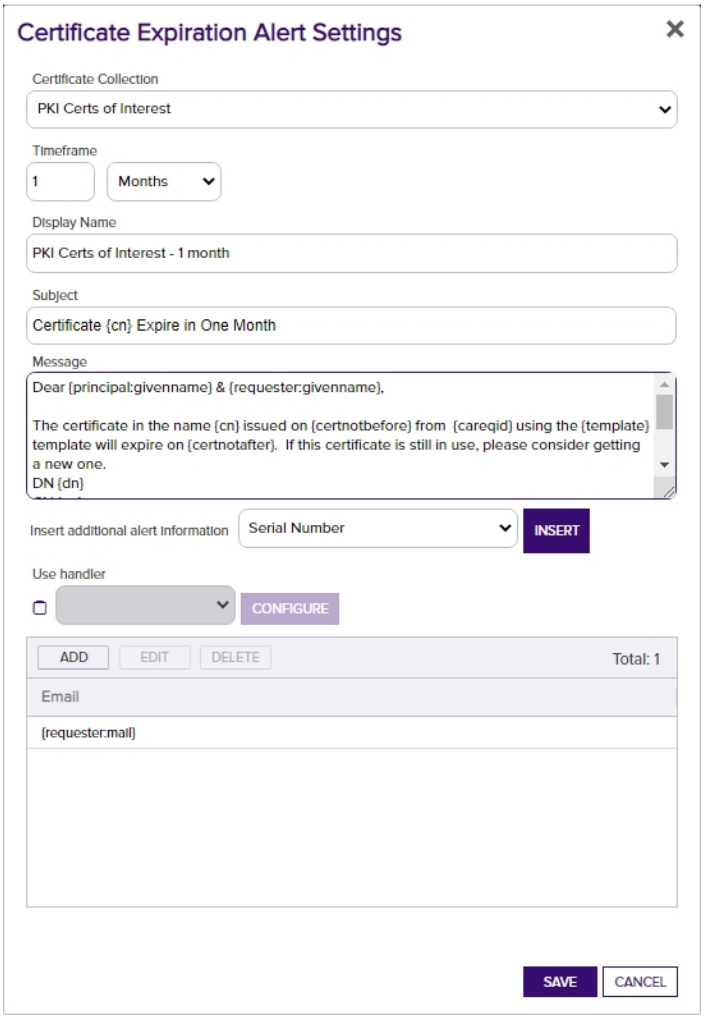


Figure 126: Create a New Expiration Alert

4. In the **Timeframe** fields, select the warning timeframe by defining a number for either days, weeks, or months for the alert. For example, if you select three weeks, the expiration alerts will be sent automatically three weeks ahead of certificate expiration.

Note: When the alert is stored in the database, weeks are converted to 7 days and months are converted to 30 days.

Example: When alerts run, the alert engine reports on all the certificates expiring within the next X days (e.g. 30 days) from the execution time that have not previously been reported on. This means that if the alerts run daily and have been running daily regularly

for some time, only a single day of expiring certificates will be reported on by any given alert run.
For example, say you create a new alert that has never run before for collection A and set it to 30 days. You configure it to run daily at 5:00 am. The alert runs for the first time at 5:00 am on July 1st. All the certificates in collection A that will expire between July 1st at 12:oo am UTC and July 31 at 12:00 am UTC will be alerted on. The next day when the alert runs again at 5:00 am on July 2nd, the certificates in collection A expiring between July 31st at 12:00 am UTC and August 1st at 12:00 am UTC will be alerted on.

If alerts are missed for a period of time (due to an outage, for example), the next run of the alerts will check the previous successful run date for the alerts and report on certificates expiring X days from that outage window. For example, using the collection A alert referenced above, say an outage caused the alerts not to run on August 1 and August 2. On August 3, the alert would run again at 5:00 am, and the certificates in collection A expiring between August 30th at 12:00 am UTC and September 2nd at 12:00 am UTC would be alerted on.

5. In the **Display Name** field, enter a name for the alert. This name appears in the list of expiration alerts in the Management Portal.

6. In the **Subject** field, enter a subject line for the email message that will be delivered when the alert is triggered. You can use substitutable special text in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {cn} in the alert definition and each alert generated at processing time will contain the specific common name of the given certificate instead of the variable {cn}.

   To add substitutable special text to the subject line; place your cursor where you would like the text to appear on the subject line, select the appropriate variable from the *Insert special text* dropdown, and click **Insert**. Alternately, type the special text variable enclosed in curly braces (e.g. {cn}).

7. In the **Message** box, enter the body of the email message that will be delivered when the alert is triggered. You can use the *Insert special text* dropdown below the message window to add substitutable special text to the message. The metadata that appears in the dropdown will depend upon the custom metadata you have defined (see Certificate Metadata on page 710). Place your cursor where you would like the text to appear, select the appropriate variable from the dropdown, and click **Insert**. Alternately, you can type the special text variable enclosed in curly braces directly. In addition to the substitutable special text fields available in the dropdown, you can also build your own substitutable fields for the principal and/or requester based on string values from the user or computer Active Directory record. See Table 4: Substitutable Special Text for Expiration Alerts. If desired, you can format the message body using HTML. For example, you could place certificate detail information into a table by replacing this text:

   DN: {dn}
   CN: {cn}

UPN: {upn}
Thumbprint: {thumbprint}
Serial Number: {serial}

With this HTML code:

```
<table>
<tr><td>DN:</td><td>{dn}</td></tr>
<tr><td>CN:</td><td>{cn}</td></tr>
<tr><td>UPN:</td><td>{upn}</td></tr>
<tr><td>Thumbprint:</td><td>{thumbprint}</td></tr>
<tr><td>Serial Number:</td><td>{serial}</td></tr>
</table>
```

8. Check the **Use handler** box if you would like the alert to trigger an event handler at processing time, select the appropriate handler in the dropdown, and click the **Configure** button to configure the event handler. See Event Handler Registration on page 739 for more information on using event handlers, and Adding PowerShell Handlers to Alerts on page 223 for more information about using PowerShell Handlers.

9. In the Recipients section of the page, click **Add** to add a recipient to the alert. Each alert can have multiple recipients. Recipients should be added one at a time. You can enter specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. The built-in variable can be selected in the Recipient dialog **Use a variable from the certificate** dropdown. In addition, you can type a special text variable enclosed in curly braces in the Email field if you have, for example, a metadata field that contains an email address.

   Keyfactor Command sends SMS (text) messages by leveraging the email to text gateways that many major mobile carriers provide. Check with your carrier for specific instructions. Keyfactor has tested that AT&T can be addressed using 10-digit-number@txt.att.net (e.g. 4155551212@txt.att.net) and Verizon can be addressed using 10-digit-number@vtext.com (e.g. 2125551212@vtext.com). T-Mobile can be addressed using 10-digit-number@tmomail.net (e.g. 2065551212@tmomail.net), but functionality can be spotty. Reliability of alerting via this method depends on the reliability of the carrier's gateways.

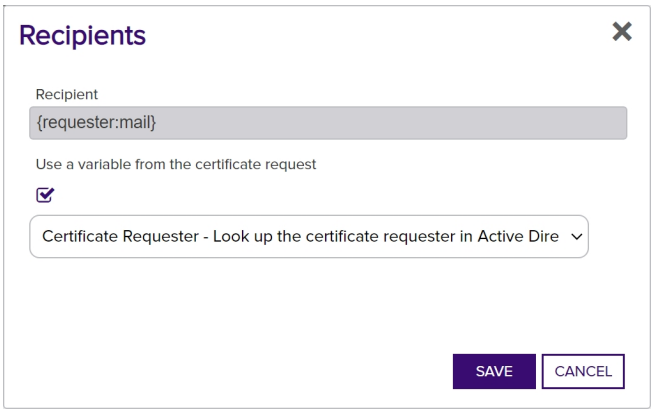


Figure 127: Expiration Alerts Recipients

10. Click **Save** to save your expiration alert, or your changes.

Copying an Expiration Alert

You may use the copy operation to create multiple similar alerts—for example, several alerts for the same certificate collection but with different warning timeframes.

1. In the Management Portal, browse to *Alerts > Expiration*.
2. On the Expiration Alerts page, highlight the row in the expiration alerts grid and click **Copy** at the top of the grid, or from the right click menu.
3. The Certificate Expiration Alert Settings dialog will pop-up with the details from the selected alert. The display name field will have - *Copy* tagged to the end of it to indicate it is a new alert. You may modify the alert as needed and click **Save** to add the new alert, or **Cancel** to cancel the operation.

Deleting an Expiration Alert

You may delete one expiration alert at a time.

1. In the Management Portal, browse to *Alerts > Expiration*.
2. On the Expiration Alerts page, highlight the row in the Expiration Alerts grid and click **Delete** at the top of the grid, or from the right click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Configuring an Expiration Alert Schedule

After adding your desired Certificate Expiration Alerts, you may configure an alert schedule.

1. In the Management Portal, browse to *Alerts > Expiration*.
2. On the Expiration Alerts page, click the **Configure** button at the top of the Expiration Alerts page to configure a monitoring execution schedule. This will apply for all the expiration alerts.

This defines the frequency with which alerts are sent. This type of alert is scheduled for daily delivery at a specified time.

Example: When alerts run, the alert engine reports on all the certificates expiring within the next X days (e.g. 30 days) from the execution time that have not previously been reported on. This means that if the alerts run daily and have been running daily regularly for some time, only a single day of expiring certificates will be reported on by any given alert run.
For example, say you create a new alert that has never run before for collection A and set it to 30 days. You configure it to run daily at 5:00 am. The alert runs for the first time at 5:00 am on July 1st. All the certificates in collection A that will expire between July 1st at 12:oo am UTC and July 31 at 12:00 am UTC will be alerted on. The next day when the alert runs again at 5:00 am on July 2nd, the certificates in collection A expiring between July 31st at 12:00 am UTC and August 1st at 12:00 am UTC will be alerted on.

If alerts are missed for a period of time (due to an outage, for example), the next run of the alerts will check the previous successful run date for the alerts and report on certificates expiring X days from that outage window. For example, using the collection A alert referenced above, say an outage caused the alerts not to run on August 1 and August 2. On August 3, the alert would run again at 5:00 am, and the certificates in collection A expiring between August 30th at 12:00 am UTC and September 2nd at 12:00 am UTC would be alerted on.

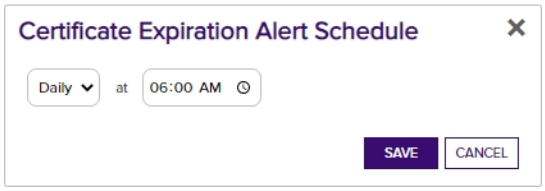


Figure 128: Expiration Alert Schedule

Testing Expiration Alerts

Once the alerts are configured, you may run a test of all, or selected, alerts to see if they are configured correctly.

1. In the Management Portal, browse to *Alerts > Expiration*.
2. On the Expiration Alerts page, either highlight one row in the expiration alert grid and click the **Test** button at the top of the grid or click the **Test All** button at the top of the grid to test all the alerts.
3. In the Expiration Alert Test dialog in the Alert Parameters section, select a **Start Date** and **End Date** for testing. You can use this option to simulate running the alerts a month from now instead of today, for example, or put in a broad date range to be sure you pick up some expiring certificates for testing purposes.

Example: Say you had experienced an outage and alerts that normally run daily at 5:00 pm did not run for two days. You wanted to test to see what to expect of the alerts once the system was up and running again. You are running your test on August 3rd for an alert that's configured to report at 30 days for collection A. The alert last ran on July 31. This means the alert has a *Previous Evaluation Date* of July 31. When running your test, set the **Start Date** to July 31st to match the *Previous Evaluation Date*. Set the **End Date** to the current date, August 2nd in this example, to simulate the results when the alerts are run today. The test results will include up to 100 certificates in collection A expiring between August 30th at 12:00 am UTC and September 1st at 12:00 am UTC.

4. In the Expiration Alert Test dialog in the Alert Parameters section, click the toggle button for **Send Alerts** if you would like to deliver email messages as part of the test.

5. Click the **Generate** button to begin generating alerts. Depending on the number of certificates to process, this may take a few seconds.

6. In the Expiration Alert Test dialog in the Alert Data and Alert Message sections, you can review the certificates found to confirm that the expected certificates are appearing and that the substitutable special text is being replaced as expected. Scroll through the alerts using the **First**, **Previous**, **Next** and **Last** buttons at the bottom of the dialog. The number of alerts generated will display between the navigation buttons.

Note: You may see fewer alerts than you have certificates expiring in the selected time window for the certificate collection if you enabled one of the options to ignore duplicate certificates on the certificate collection (see Saving Search Criteria as a Collection on page 42).

Tip: Alerts are generated when a certificate has expired or is approaching expiration as defined by the timeframe configured in the alert.
By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Expiration Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602 in the *Keyfactor Command Reference Guide*). If more than 100 alerts are generated, no email messages will be sent and you'll have the opportunity to view the first 100 alerts generated.

If you're using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message written, certificates renewed) when the test is run. This is true whether or not you click the *Send Alerts* toggle.

Note: HTML does not render in the alert viewer.

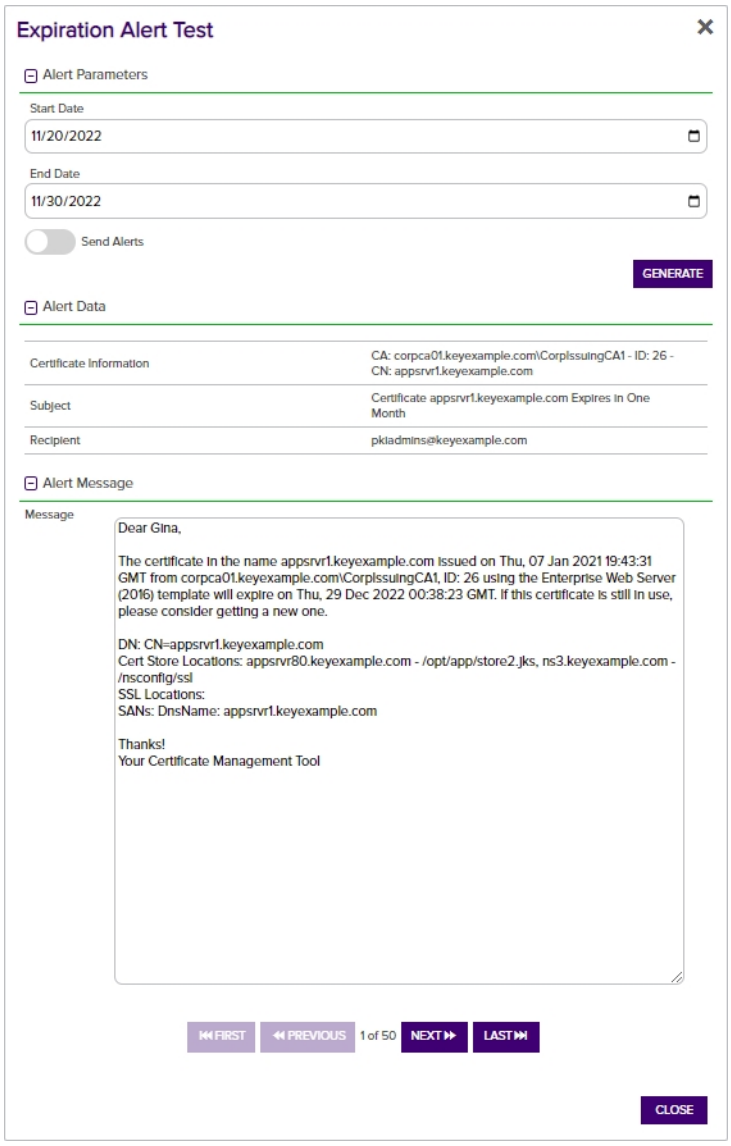


Figure 129: Expiration Alert Test

Refer to the following table for a complete list of the substitutable special text that can be used to customize alert messages.

Table 4: Substitutable Special Text for Expiration Alerts

Variable	Name	Description
{certemail}	Email Address in Certificate	Email address contained in the certificate, if present
{cn}	Common Name	Common name contained in the certificate

Variable	Name	Description
{dn}	Distinguished Name	Distinguished name contained in the certificate
{certnotbefore}	Issue Date	Validity date of the certificate
{certnotafter}	Expiration Date	Expiration date of the certificate
{issuerDN}	Issuer DN	Distinguished name of the certificate's issuer
{locations:certstore}	Certificate Store Locations	The server and path location to the certificate store (s) where the certificate resides, if any, for certificates found in certificate stores (e.g. server1.keyexample.com – /opt/test/mystore.jks)
{principal:mail}	Principal's Email	Email address retrieved from Active Directory of the user whose UPN is contained in the SAN field of the certificate, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{principal:givenname}	Principal's First Name	First name retrieved from Active Directory of the user whose UPN is contained in the SAN field of the certificate, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{principal:sn}	Principal's Last Name	Last name retrieved from Active Directory of the user whose UPN is contained in the SAN field of the certificate, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{principal:displayname}	Principal's Display Name	Display name retrieved from Active Directory of the user whose UPN is contained in the SAN field of the certificate, if present **Note:** This substitutable special text token is

Variable	Name	Description
		only supported in environments using Active Directory as an identity provider.
{requester}	Requester	The user account that requested the certificate from the CA, in the form *DOMAIN\username*
{requester:mail}	Requester's Email	Email address retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:givenname}	Requester's First Name	First name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:sn}	Requester's Last Name	Last name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:displayname}	Requester's Display Name	Display name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{careqid}	Issuing CA / Request ID	A string containing the Issuing CA name and the certificate's Request ID from the CA

Variable	Name	Description
{serial}	Serial Number	The serial number of the certificate
{locations:ssl}	SSL Locations	The server location(s) where the certificate resides, if any, for certificates synchronized using SSL synchronization
{san}	Subject Alternative Name	Subject alternative name(s) contained in the certificate
{template}	Template Name	Name of the certificate template used to create the certificate
{templateshortname}	Template Short Name	Short name (often the name with no spaces) of the certificate template used to create the certificate
{thumbprint}	Thumbprint	The thumbprint (hash) of the certificate
{upn}	User Principal Name	The user principal name (UPN) contained in the subject alternative name (SAN) field of the certificate, if present (e.g. username@keyexample.com)
{metadata: Email-Contact}	Email-Contact	Example of a custom metadata field
{principal:field}	String Value from AD	Locates the object in Active Directory identified by the UPN in the certificate (if present), and substitutes the contents of the attribute named by *field*. For example: • {principal:department} • {principal:sAMAccountName} • {principal:manager} • {principal:co} **Note:** This substitutable special text field is partially user defined—you pick the field out of AD to include—and is therefore not available in the *Insert special text* dropdown; it needs to be typed manually. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:field}	String Value	Locates the object in Active Directory identified by

Variable	Name	Description
	from AD	the user or computer account that requested the certificate from the CA, and substitutes the contents of the attribute named by *field*. For example, for users: • {requester:department} • {requester:sAMAccountName} For computers: • {requester:operatingSystem} • {requester:location} • {requester:managedBy} **Note:** This substitutable special text field is partially user defined—you pick the field out of AD to include—and is therefore not available in the *Insert special text* dropdown; it needs to be typed manually. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.

2.1.6.2 Pending Certificate Request Alerts

Pending certificate request alerts are used to send email notifications to certificate administrators when a new certificate request that requires approval based on policy on the CA is generated. The alerts can be customized to provide detailed information about the certificate requests.

Important: These alerts are **not** used to provide email alerts or run event handlers for certificate requests that require approval based on policies configured in Keyfactor Command workflows. Pending request notification for requests handled by Keyfactor Command workflow are configured within the workflow (see Adding, Copying or Modifying a Workflow Definition on page 237).

Pending certificate requests are generated, for the most part, based on templates that are configured to require manager approval at the CA level.

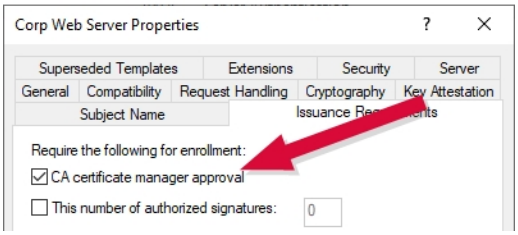


Figure 130: Certificate Template Requiring Manager Approval

The functionality of pending alerts for certificates requested within Keyfactor Command has been largely replaced by the new Keyfactor Command workflow added in Keyfactor Command version 10 (see Workflow on page 229). When alerting with Keyfactor Command workflow, templates do not need to be configured to require manager approval. This is because the approval handling is fully controlled within Keyfactor Command. In fact, templates generally should not be configured to require manager approval when using Keyfactor Command workflow, since this would generally require approval both at the Keyfactor Command level and at the CA level, depending on workflow configuration.

Pending alerts are retained for use in these scenarios:

- For customers not wishing to make use of Keyfactor Command workflow.
- For customers still in the process of migrating from CA-based workflow to Keyfactor Command workflow.
- For certificates requested outside of Keyfactor Command using templates that require manager approval.

Note: Pending request alerts are supported only for Microsort CAs and select CA gateways. This feature is not supported for EJBCA CAs.

Pending Request Alert Operations

Pending certificate request alerts are designed to send an email notification to certificate approvers when a certificate request is received that requires approval based on policy on the CA. Pending request alerts can also be sent to the original certificate requesters alerting them that their certificate requests have been sent.

Important: These alerts are **not** used to provide email alerts or run event handlers for certificate requests that require approval based on policies configured in Keyfactor Command workflows. Pending request notification for requests handled by Keyfactor Command workflow are configured within the workflow (see Adding, Copying or Modifying a Workflow Definition on page 237).

Pending Request Alert operations include:

- Creating, editing or deleting a pending alert
- Configuring an alert schedule
- Copying alerts to create similar alerts for different recipients or situations
- Testing alerts

Tip: In order to be used for PFX enrollment, a template that requires manager approval must be configured with private key retention to allow the private key generated for the request to be downloaded with the certificate after the certificate request is approved (see Certificate Template Operations on page 381).

Adding or Modifying a Pending Request Alert

1. In the Management Portal, browse to *Alerts > Pending Request*.
2. On the Pending Certificate Request Alerts page, click **Add** from the top menu to create a new alert, or **Edit** from either the top or right click menu, to modify an existing one.
3. In the Pending Request Alert Settings dialog, select your **Certificate Template** (or select **All Templates**) in the first dropdown.

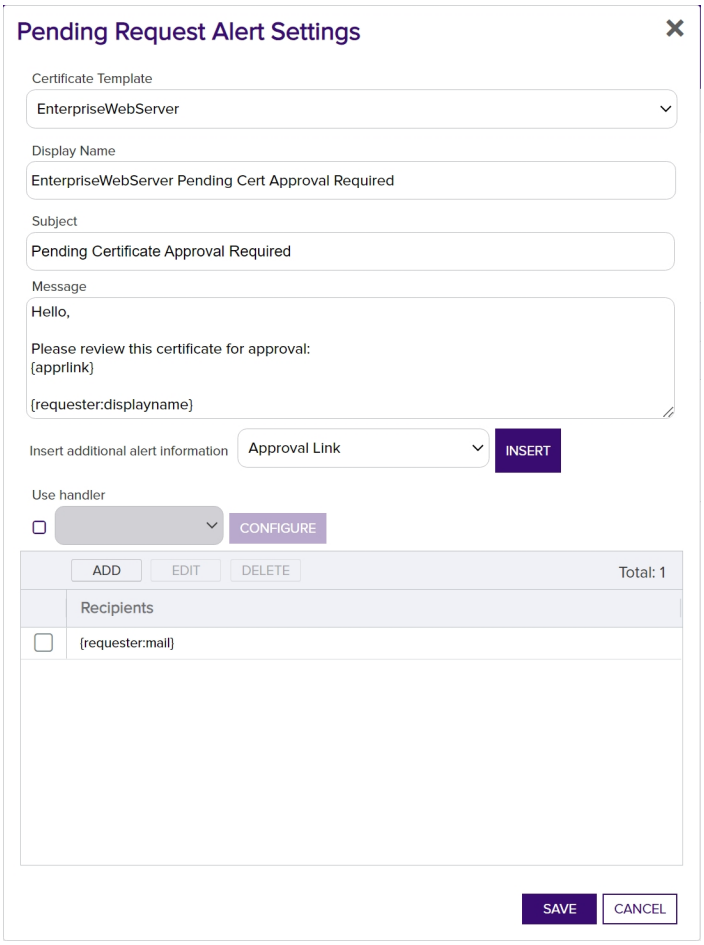


Figure 131: Create a New Pending Request Alert

4. In the **Display Name** field, enter a name for the alert. This name appears in the pending request alerts grid in the Management Portal.

5. In the **Subject** field, enter a subject line for the email message that will be delivered when the alert is triggered. You can use substitutable special text in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.

   To add substitutable special text to the subject line; place your cursor where you would like the text to appear on the subject line, select the appropriate variable from the *Insert special text* dropdown, and click **Insert**. Alternately, type the special text variable enclosed in curly braces (e.g. {cn}).

6. In the **Message** box, enter the body of the email message that will be delivered when the alert is triggered. You can use the **Insert special text** dropdown below the message window to add substitutable special text to the message. The metadata that appears in the dropdown will depend upon the custom metadata you have defined (see Certificate Metadata on page 710). Place your cursor where you would like the text to appear, select the appropriate variable from the dropdown, and click **Insert**. Alternately, you can type the special text variable enclosed in curly braces directly. In addition to the substitutable special text fields available in the dropdown, you can also build your own substitutable fields for the requester based on string values from the user or computer Active Directory record. See Table 5: Substitutable Special Text for Pending Request Alerts. If desired, you can format the message body using HTML. For example, you could place the certificate detail information into a table by replacing this text:

   ```
   CN: {rcn}
   DN: {rdn}
   SAN: {san}
   ```

   With this HTML code:

   ```
   <table>
   <tr><td>CN:</td><td>{rcn}</td></tr>
   <tr><td>DN:</td><td>{rdn}</td></tr>
   <tr><td>SAN:</td><td>{san}</td></tr>
   </table>
   ```

7. The **Approval Link** substitutable special text field is an important one to include in your alert intended for the administrator responsible for approving or denying the certificate request. This provides a link in the email message that the administrator can click to be taken to an approve/deny page for the certificate in the Management Portal to either approve or deny the request. This certificate-specific approval page cannot be directly accessed within the Management Portal (though you can approve certificate requests in the Management Portal from the Certificate Requests page (see Certificate Requests on page 163).

8. Check the **Use handler** box if you would like the alert to trigger an event handler at processing time, select the appropriate handler in the dropdown, and click the **Configure** button to configure the event handler. See Event Handler Registration on page 739 for more information on using event handlers, and Adding PowerShell Handlers to Alerts on page 223 for more information about using PowerShell Handlers.

9. In the Recipients section of the page, click **Add** to add a recipient to the alert. Each alert can have multiple recipients. Recipients should be added one at a time. You can enter specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. The built-in variable can be selected in the Recipient dialog **Use a variable from the certificate request** dropdown. In addition, you can type a special text variable enclosed in curly braces in the Email field if you have, for example, a metadata field that contains an email address.

Keyfactor Command sends SMS (text) messages by leveraging the email to text gateways that many major mobile carriers provide. Check with your carrier for specific instructions. Keyfactor has tested that AT&T can be addressed using 10-digit-number@txt.att.net (e.g. 4155551212@tx-t.att.net) and Verizon can be addressed using 10-digit-number@vtext.com (e.g. 2125551212@v-text.com). T-Mobile can be addressed using 10-digit-number@tmomail.net (e.g. 2065551212@tmomail.net), but functionality can be spotty. Reliability of alerting via this method depends on the reliability of the carrier's gateways.

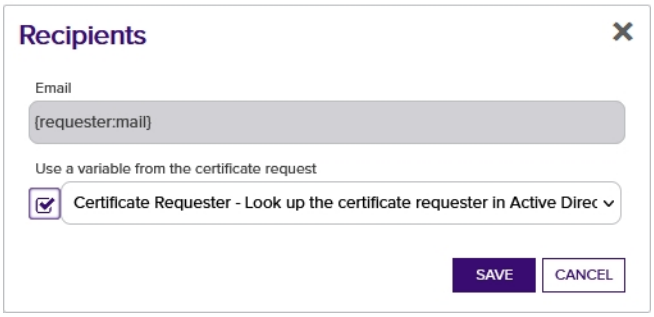


Figure 132: Pending Request Alerts Recipients

10. Click **Save** to save your pending request alert.

Copying a Pending Request Alert

You may use the copy operation to create multiple similar alerts—for example, one to the requester of the certificate and one to the administrator(s) responsible for approving it.

1. In the Management Portal, browse to *Alerts > Pending Request*.
2. On the Pending Certificate Request Alerts page, highlight the row in the alerts grid and click **Copy** at the top of the grid, or from the right click menu.
3. The Pending Request Alert Settings dialog will pop-up with the details from the selected alert. The display name field will have - *Copy* tagged to the end of it to indicate it is a new alert. You may modify the alert as needed and click **Save** to add the new alert, or **Cancel** to cancel the operation.

Deleting a Pending Request Alert

1. In the Management Portal, browse to *Alerts > Pending Request*.
2. On the Pending Certificate Request Alerts page, highlight the row in the alerts grid and click **Delete** at the top of the grid, or from the right click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Configuring a Pending Request Alert Schedule

After adding your desired pending request alerts, you may configure a schedule to send the alerts.

1. In the Management Portal, browse to *Alerts > Pending Request*.
2. On the Pending Certificate Request Alerts page, click the **Configure** button at the top of the Pending Request Alerts page to open the **Pending Certificate Request Alert Schedule** dialog and configure a monitoring execution schedule. This defines the frequency with which alerts are sent. You can choose to schedule the alerts for:

- **Daily** delivery at a specified time
- An **Interval** of anywhere from every 1 minute to every 12 hours
- Turn **Off** a previously configured schedule

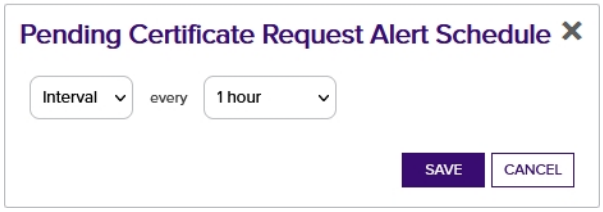


Figure 133: Pending Request Alert Schedule

Testing Pending Request Alerts

Once the alerts are configured, you may run a test of all or selected alerts to see if they are configured correctly.

1. In the Management Portal, browse to *Alerts > Pending Request*.
2. On the Pending Certificate Request Alerts page, either highlight one row in the pending request alerts grid and click the **Test** button at the top of the grid or click the **Test All** button at the top of the grid to test all the alerts.
3. In the Pending Alert Test dialog in the Alert Parameters section, click the toggle button for **Send Alerts** , if you would like to deliver email messages as part of the test.
4. Click the **Generate** button to begin generating alerts. Depending on the number of certificate requests to process, this may take a few seconds.
5. In the Pending Alert Test dialog in the Alert Data and Alert Message sections, you can review the certificate requests found to confirm that the expected requests are appearing and that the substitutable special text is being replaced as expected. Scroll the **First**, **Previous**, **Next** and **Last** buttons at the bottom of the dialog. The number of alerts generated will display between the navigation buttons.

Tip: Alerts are generated for all certificate requests that have not previously been alerted on, unless the system has been configured to send multiple alerts per request. By default, one alert is sent to each recipient for any given request. The number of alerts to send for a given request is configurable with the *Pending Alert Max Reminders* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If a

certificate remains in a pending state after the configured number of alerts has been sent, no further alerts will be sent.
By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Pending Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If more than 100 alerts are generated, no email messages will be sent and you'll have the opportunity to view the first 100 alerts generated.

If you're using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message) when the test is run. This is true whether or not you click the *Send Alerts* toggle.

Note: HTML does not render in the alert viewer.

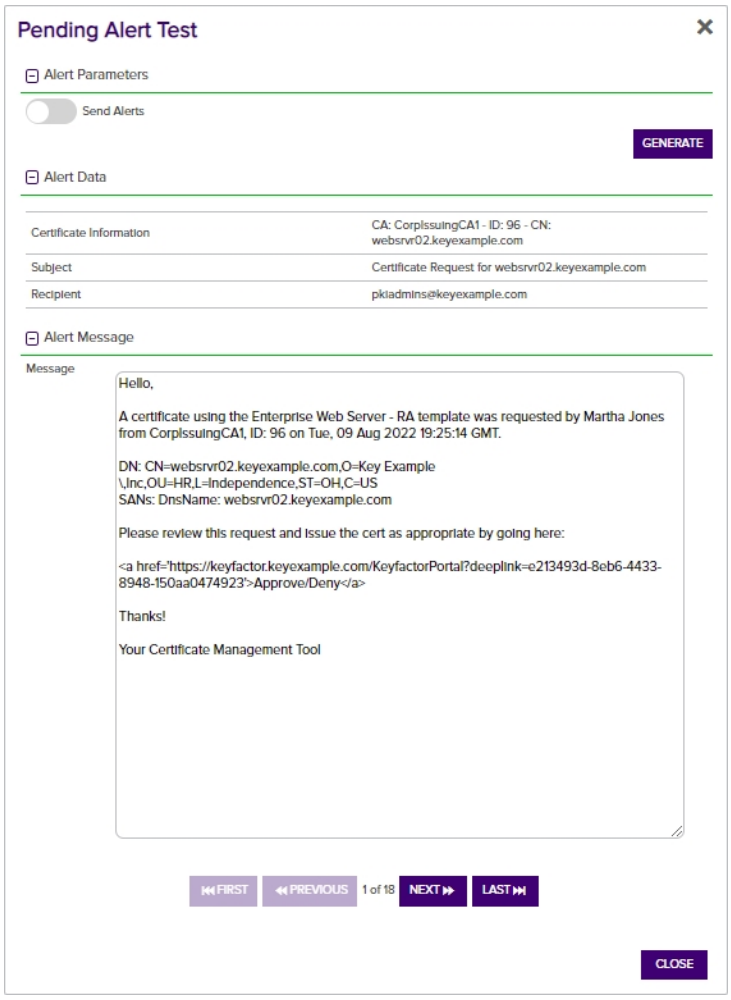


Figure 134: Pending Alert Test

Refer to the following table for a complete list of the substitutable special text that can be used to customize alert messages.

Table 5: Substitutable Special Text for Pending Request Alerts

Variable	Name	Description
{apprlink}	Approval Link	Link pointing to the certificate-specific approval page in the Management Portal where the person responsible for approving the request can go to approve or deny the request
{reqid}	CMS Request Id	The request ID for the certificate as stored in the Keyfactor Command database. This is not the same as the request ID issued by the CA.
{rcn}	Requested Common Name	Common name contained in the certificate request
{rdn}	Requested Distinguished Name	Distinguished name contained in the certificate request
{requester}	Requester	The user account that requested the certificate from the CA, in the form *DOMAIN\username*
{requester:mail}	Requester's Email	Email address retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:givenname}	Requester's First Name	First name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:sn}	Requester's Last Name	Last name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is

Variable	Name	Description
		only supported in environments using Active Directory as an identity provider.
{requester:displayname}	Requester's Display Name	Display name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{careqid}	Issuing CA / Request ID	A string containing the Issuing CA name and the certificate's Request ID from the CA
{san}	Subject Alternative Name	Subject alternative name(s) contained in the certificate request. There are four possible sources for the SANs that appear here: • For CSR enrollment, the original SANs included in the CSR. • Any SANs added through the Keyfactor Command Management Portal. For CSR enrollment, these take the place of the SANs in the CSR if the ATTRIBUTESUBJECTALTNAME2 option is enabled on the CA. See CSR Enrollment on page 136. • A SAN matching the CN added automatically during enrollment as a result of setting the RFC 2818 compliance flag in the CA configuration. See Adding or Modifying a CA Record on page 354. For PFX enrollment, the user has the option of editing this entry at enrollment time; entry of something is required. • A SAN matching the CN added automatically by the Keyfactor Command policy module on the CA if the Keyfactor Command RFC 2818 Policy Handler is enabled, if one was not included in the CSR or added manually. See Installing the Keyfactor CA Policy Module Handlers on page 2846 in the *Keyfactor Command*

Variable	Name	Description
		Server Installation Guide.
{subdate}	Submission Date	Date the certificate request was submitted
{template}	Template Name	Name of the certificate template used to create the certificate request
{templateshortname}	Template Short Name	Short name (often the name with no spaces) of the certificate template used to create the certificate request
{metadata: Email-Contact}	Email-Contact	Example of a custom metadata field
{requester:field}	String Value from AD	Locates the object in Active Directory identified by the user or computer account that requested the certificate from the CA, and substitutes the contents of the attribute named by *field*. For example, for users: • {requester:department} • {requester:sAMAccountName} For computers: • {requester:operatingSystem} • {requester:location} • {requester:managedBy} **Note:** This substitutable special text field is partially user defined—you pick the field out of AD to include—and is therefore not available in the *Insert special text* dropdown; it needs to be typed manually. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.

2.1.6.3 Issued Certificate Request Alerts

Issued certificate request alerts are used to send email notifications to certificate requesters, or other relevant parties, when a new certificate is issued through any CA that syncs to Keyfactor Command. The alerts can be customized to provide detailed information about the certificates.

Note: Because Issued Certificate Request Alerts are sent for any CAs synced to Keyfactor Command, it is recommended that any CAs are synced first and then the Issued Certificate Request Alerts set up afterward to avoid a lot of unnecessary emails, upon syncing.

Issued Request Alert Operations

An issued certificate request alert is designed to send an email notification to a certificate requester when a certificate request he or she made using a certificate template that required manager approval is approved.

Issued Request Alert operations include: creating, editing or deleting an issued request alerts, configuring an alert schedule, and copying alerts to create similar alerts for different recipients or collections.

The issued alert handler runs immediately when an enrollment is approved within the Keyfactor Command platform and also runs via a schedule to pick up any approvals done outside of Keyfactor Command.

Adding or Modifying an Issued Request Alert

1. In the Management Portal, browse to *Alerts > Issued Request*.
2. On the Issued Certificate Request Alerts page, click **Add** from the top menu to create a new alert, or **Edit** ,from either the top or right click menu, to modify an existing one.
3. In the Issued Certificate Alert Settings dialog, select your **Certificate Template** (or select **All Templates**) in the first dropdown.

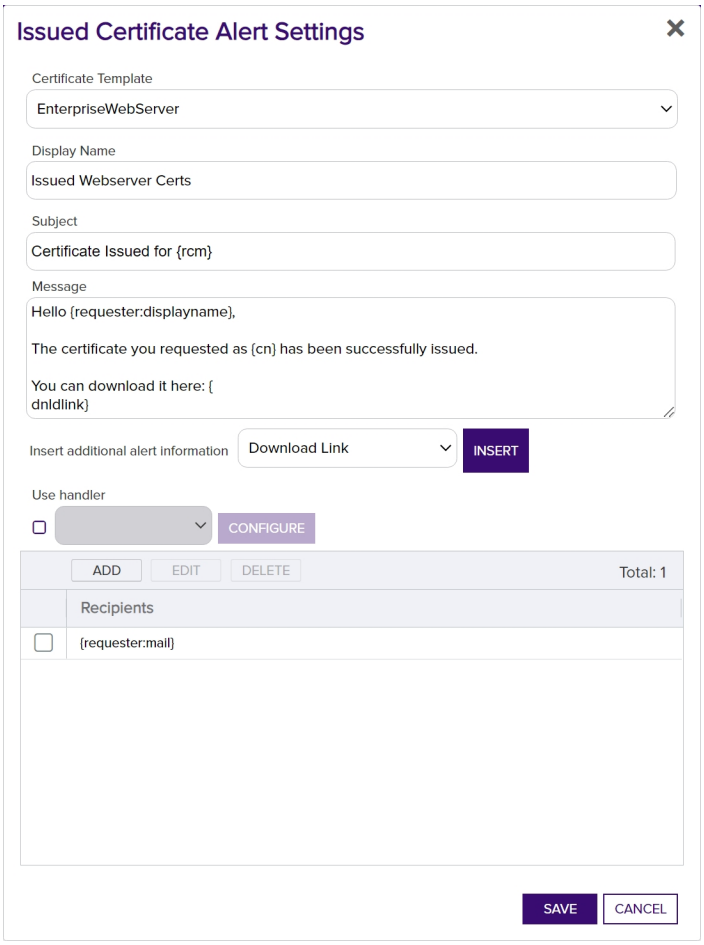


Figure 135: Create a New Issued Certificate Alert

4. In the **Display Name** field, enter a name for the alert. This name appears in the list of issued certificate alerts in the Management Portal.

5. In the **Subject** field, enter a subject line for the email message that will be delivered when the alert is triggered. You can use substitutable special text in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {cn} in the alert definition and each alert generated will contain the specific common name of the given certificate instead of the variable {cn}.

   To add substitutable special text to the subject line; place your cursor where you would like the text to appear on the subject line, select the appropriate variable from the *Insert special text* dropdown, and click **Insert**. Alternately, type the special text variable enclosed in curly braces (e.g. {cn}).

6. In the **Message** box, enter the body of the email message that will be delivered when the alert is triggered. You can use the **Insert special text** dropdown below the message window to add substitutable special text to the message. The metadata that appears in the dropdown will depend upon the custom metadata you have defined (see Certificate Metadata on page 710). Place your cursor where you would like the text to appear, select the appropriate variable from the dropdown, and click **Insert**. Alternately, you can type the special text variable enclosed in curly braces directly. In addition to the substitutable special text fields available in the dropdown, you can also build your own substitutable fields for the requester based on string values from the user or computer Active Directory record. See Substitutable Special Text for Issued Certificate Alerts on page 194. If desired, you can format the message body using HTML. For example, you could place the certificate detail information into a table by replacing this text:

   ```
   Serial Number: {serial}
   Thumbprint: {thumbprint}
   SANs: {san}
   App Owner: {metadata:AppOwnerFirstName} {metadata:AppOwnerLastName}
   ```

   With this HTML code:

   ```
   <table>
   <tr><td>Serial Number: </td><td>{serial}</td></tr>
   <tr><td>Thumbprint: </td><td>{thumbprint}</td></tr>
   <tr><td>SANs: </td><td>{san}</td></tr>
   <tr><td>App Owner: </td><td>{metadata:AppOwnerFirstName} {metadata:Ap-
   pOwnerLastName}</td></tr>
   </table>
   ```

7. The **Download Link** substitutable special text field is an important one to include in your alert intended for the requester of the certificate or the person responsible for installing the certificate. This provides a link in the email message that the user can click to be taken to the Keyfactor Command Management Portal to download the certificate.

   Tip: If the users who will receive the issued alerts do not have global *Read* permissions for *Certificates*, they will not be able to use the built-in download link. To resolve this, you can build a custom download link as follows:

   a. If you do not already have a *My Certificates* collection, create one using the %ME% special value with a search string of:

      ```
      NetBIOSRequester -eq "%ME%"
      ```

   b. In the Management Portal, browse to the *My Certificates* collection page and look in the browser's address bar at the end of the URL for the number that has been assigned to the collection. For example, the following URL points to collection 9:

      ```
      https://keyfactor.keyexample.com/KeyfactorPortal/CertificateCollection/Edi
      t?cid=9
      ```

c. Grant the users who will receive the issued alerts *Read* permissions on the *My Certificates* collection (see Certificate Collection Permissions on page 627).

d. In the message body of the issued alert, create a link that looks like the following, where keyfactor.keyexample.com is the name of your Keyfactor Command server and ID is the correct collection ID for your *My Certificates* collection (e.g. 9):

```
<a
href="https://
keyfactor.keyexample.com
/KeyfactorPortal/CertificateCollection/Edit?cid=ID&query=Thumbprint+-
eq+%22{thumbprint}%22">Download Now</a>
```

8. Check the **Use handler** box if you would like the alert to trigger an event handler at processing time, select the appropriate handler in the dropdown, and click the **Configure** button to configure the event handler. See Event Handler Registration on page 739 for more information on using event handlers, and Adding PowerShell Handlers to Alerts on page 223 for more information about using PowerShell Handlers.

9. In the Recipients section of the page, click **Add** to add a recipient to the alert. Each alert can have multiple recipients. Recipients should be added one at a time. You can enter specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. There are three built-in variables that can be selected in the Recipient dialog **Use a variable from the certificate request** dropdown. In addition, you can type a special text variable enclosed in curly braces in the Email field if you have, for example, a metadata field that contains an email address.

   Keyfactor Command sends SMS (text) messages by leveraging the email to text gateways that many major mobile carriers provide. Check with your carrier for specific instructions. Keyfactor has tested that AT&T can be addressed using 10-digit-number@txt.att.net (e.g. 4155551212@txt.att.net) and Verizon can be addressed using 10-digit-number@vtext.com (e.g. 2125551212@vtext.com). T-Mobile can be addressed using 10-digit-number@tmomail.net (e.g. 2065551212@tmomail.net), but functionality can be spotty. Reliability of alerting via this method depends on the reliability of the carrier's gateways.

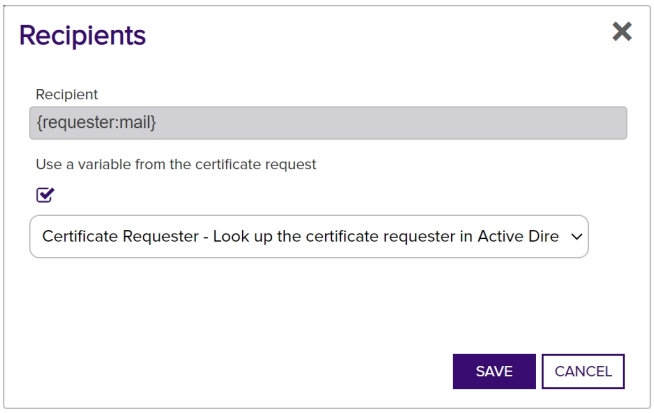


Figure 136: Issued Certificate Alerts Recipients

10. Click **Save** to save your issued certificate alert.

Copying an Issued Request Alert

You may use the copy operation to create multiple similar alerts—for example, one to the requester of the certificate and another with a different message to the person responsible for installing it.

1. In the Management Portal, browse to *Alerts > Issued Request*.
2. On the Issued Certificate Request Alerts page, highlight the row in the alerts grid and click **Copy** at the top of the grid, or from the right click menu.
3. The Issued Certificate Alert Settings dialog will pop-up with the details from the selected alert. The display name field will have - *Copy* tagged to the end of it to o indicate it is a new alert. You may modify the alert as needed and click **Save** to add the new alert, or **Cancel** to cancel the operation.

Deleting an Issued Request Alert

1. In the Management Portal, browse to *Alerts > Issued Request*.
2. On the Issued Certificate Request Alerts page, highlight the row in the alerts grid and click **Delete** at the top of the grid, or from the right click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Configuring an Issued Request Alert Schedule

After adding your desired issued alerts, you may configure a schedule to send the alerts.

1. In the Management Portal, browse to *Alerts > Issued Request*.
2. On the Issued Certificate Request Alerts page, click the **Configure** button at the top of the Issued Certificate Request Alerts page to configure a monitoring execution schedule. This defines the frequency with which alerts are sent. You can choose to schedule the alerts either

for daily delivery at a specified time or at intervals of anywhere from every 1 minute to every 12 hours. A short interval is the most common configuration.

Figure 137: Issued Alert Schedule

Refer to the following table for a complete list of the substitutable special text that can be used to customize alert messages.

Table 6: Substitutable Special Text for Issued Certificate Alerts

Variable	Name	Description
{dnldlink}	Download Link	Link pointing to the Certificate Requests page in the Keyfactor Command Management Portal where the certificate requester or the person responsible for installing the certificate can go to download the certificate. The certificate will be available only in a .cer-/.crt format (without the private key) unless private key retention has been enabled on the template (see Certificate Templates on page 379).
{certemail}	Email Address in Certificate	Email address contained in the certificate, if present
{cn}	Common Name	Common name contained in the certificate
{dn}	Distinguished Name	Distinguished name contained in the certificate
{certnotbefore}	Issue Date	Validity date of the certificate
{certnotafter}	Expiration Date	Expiration date of the certificate
{issuerDN}	Issuer DN	Distinguished name of the certificate's issuer
{principal:mail}	Principal's Email	Email address retrieved from Active Directory of the user whose UPN is contained in the SAN field of the certificate, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.

Variable	Name	Description
{principal:givenname}	Principal's First Name	First name retrieved from Active Directory of the user whose UPN is contained in the SAN field of the certificate, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{principal:sn}	Principal's Last Name	Last name retrieved from Active Directory of the user whose UPN is contained in the SAN field of the certificate, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{principal:displayname}	Principal's Display Name	Display name retrieved from Active Directory of the user whose UPN is contained in the SAN field of the certificate, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester}	Requester	The user account that requested the certificate from the CA, in the form *DOMAIN\username*
{requester:mail}	Requester's Email	Email address retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:givenname}	Requester's First Name	First name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.

Variable	Name	Description
{requester:sn}	Requester's Last Name	Last name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:displayname}	Requester's Display Name	Display name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{careqid}	Issuing CA / Request ID	A string containing the Issuing CA name and the certificate's Request ID from the CA
{serial}	Serial Number	The serial number of the certificate
{san}	Subject Alternative Name	Subject alternative name(s) contained in the certificate
{template}	Template Name	Name of the certificate template used to create the certificate
{templateshortname}	Template Short Name	Short name (often the name with no spaces) of the certificate template used to create the certificate request
{thumbprint}	Thumbprint	The thumbprint (hash) of the certificate
{upn}	User Principal Name	The user principal name (UPN) contained in the subject alternative name (SAN) field of the certificate, if present (e.g. username@keyexample.com)
{metadata:Email-Contact}	Email-Contact	Example of a custom metadata field
{principal:field}	String Value from AD	Locates the object in Active Directory identified by the UPN in the certificate (if present), and substitutes the contents of the attribute named by *field*. For example: • {principal:department}

Variable	Name	Description
		• {principal:sAMAccountName} • {principal:manager} • {principal:co} **Note:** This substitutable special text field is partially user defined—you pick the field out of AD to include—and is therefore not available in the *Insert special text* dropdown; it needs to be typed manually. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:field}	String Value from AD	Locates the object in Active Directory identified by the user or computer account that requested the certificate from the CA, and substitutes the contents of the attribute named by *field*. For example, for users: • {requester:department} • {requester:sAMAccountName} For computers: • {requester:operatingSystem} • {requester:location} • {requester:managedBy} **Note:** This substitutable special text field is partially user defined—you pick the field out of AD to include—and is therefore not available in the *Insert special text* dropdown; it needs to be typed manually. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.

2.1.6.4 Denied Certificate Request Alerts

Denied certificate request alerts are used to send email notifications to certificate requesters or other relevant parties when a certificate request that required approval is denied through Keyfactor

Command. The alerts can be customized to provide detailed information about the certificate requests.

Important: These alerts are **not** used to provide email alerts or run event handlers for certificate requests that require approval based on policies configured in Keyfactor Command workflows. Denial notification for requests handled by Keyfactor Command workflow are configured within the workflow (see Adding, Copying or Modifying a Workflow Definition on page 237).

Unlike pending certificate request alerts that are sent on a configurable schedule, denied certificate request alerts are sent immediately after the certificate request is denied through Keyfactor Command.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Denied Certificate Request Alert Operations

A denied certificate request alert is designed to send an email notification to a certificate requester when a certificate request he or she made using a certificate template that required manager approval is denied. It can include a comment from the administrator who denied the request indicating why the request was denied. From the Denied Certificate Request Alert page you can add a new alert, edit an existing one, delete an alert and copy an existing alert to form a template for a new alert.

Adding or Modifying a Denied Certificate Request Alert

1. In the Management Portal, browse to *Alerts* > *Denied Request*.
2. On the Denied Certificate Requests Alerts page, click **Add** at the top of the grid to create a new alert, or click **Edit** to modify an existing one (**Edit** is also available from the right click menu).
3. In the Denied Certificate Request Alert Settings dialog, select your Certificate Template (or select **All Templates**) in the first dropdown.

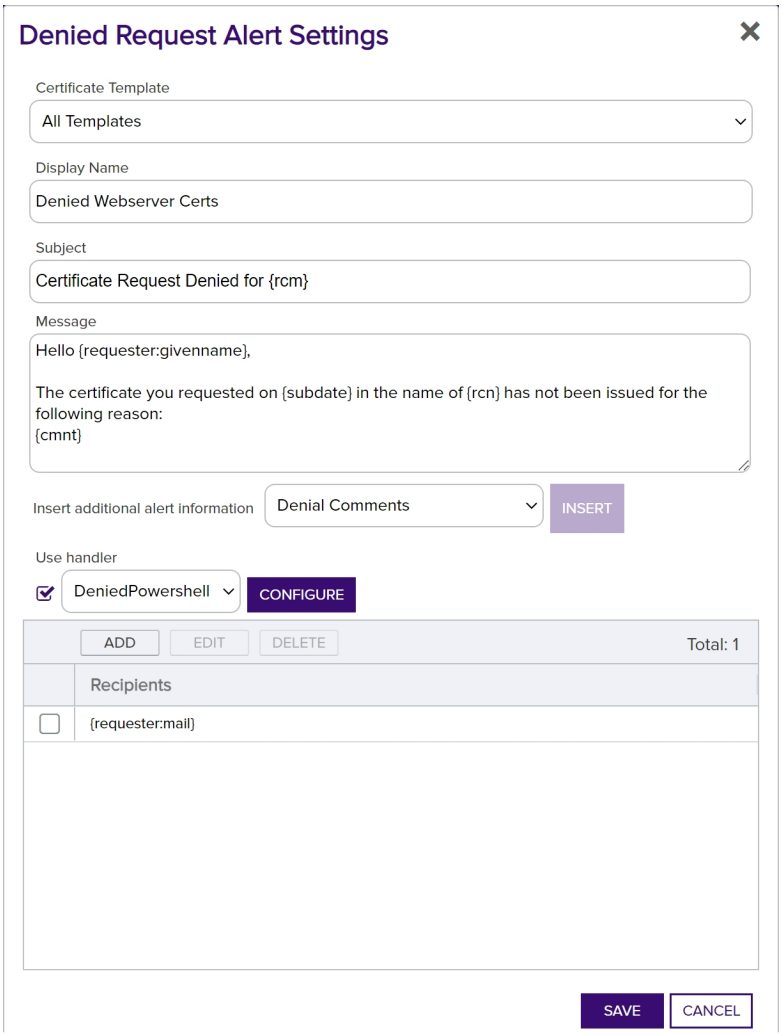


Figure 138: Create a New Denied Certificate Request Alert

4. In the **Display Name** field, enter a name for the alert. This name appears in the list of denied certificate request alerts in the Management Portal.

5. In the **Subject** field, enter a subject line for the email message that will be delivered when the alert is triggered. You can use substitutable special text in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated will contain the specific requested common name of the given request instead of the variable {rcn}.

   To add substitutable special text to the subject line; place your cursor where you would like the text to appear on the subject line, select the appropriate variable from the *Insert special text*

dropdown, and click **Insert**. Alternately, type the special text variable enclosed in curly braces (e.g. {cn}).

6. In the **Message** box, enter the body of the email message that will be delivered when the alert is triggered. You can use the **Insert special text** dropdown below the message window to add substitutable special text to the message. The metadata that appears in the dropdown will depend upon the custom metadata you have defined (see Certificate Metadata on page 710). Place your cursor where you would like the text to appear, select the appropriate variable from the dropdown, and click **Insert**. Alternately, you can type the special text variable enclosed in curly braces directly. In addition to the substitutable special text fields available in the dropdown, you can also build your own substitutable fields for the requester based on string values from the user or computer Active Directory record. See Table 7: Substitutable Special Text for Denied Certificate Request Alerts. If desired, you can format the message body using HTML.

7. The **Denial Comments** substitutable special text field is an important one to include in your alert intended for the requester of the certificate. This provides the comment the administrator made at the time he or she denied the certificate request (see Certificate Requests on page 163).

8. Check the **Use handler** box if you would like the alert to trigger an event handler at processing time, select the appropriate handler in the dropdown, and click the **Configure** button to configure the event handler. See Event Handler Registration on page 739 for more information on using event handlers, and Adding PowerShell Handlers to Alerts on page 223 for more information about using PowerShell Handlers.

9. In the Recipients section of the page, click **Add** to add a recipient to the alert. Each alert can have multiple recipients. Recipients should be added one at a time. You can enter specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. There are three built-in variables that can be selected in the Recipient dialog **Use a variable from the certificate request** dropdown. In addition, you can type a special text variable enclosed in curly braces in the Email field if you have, for example, a metadata field that contains an email address.

   Keyfactor Command sends SMS (text) messages by leveraging the email to text gateways that many major mobile carriers provide. Check with your carrier for specific instructions. Keyfactor has tested that AT&T can be addressed using 10-digit-number@txt.att.net (e.g. 4155551212@txt.att.net) and Verizon can be addressed using 10-digit-number@vtext.com (e.g. 2125551212@vtext.com). T-Mobile can be addressed using 10-digit-number@tmomail.net (e.g. 2065551212@tmomail.net), but functionality can be spotty. Reliability of alerting via this method depends on the reliability of the carrier's gateways.

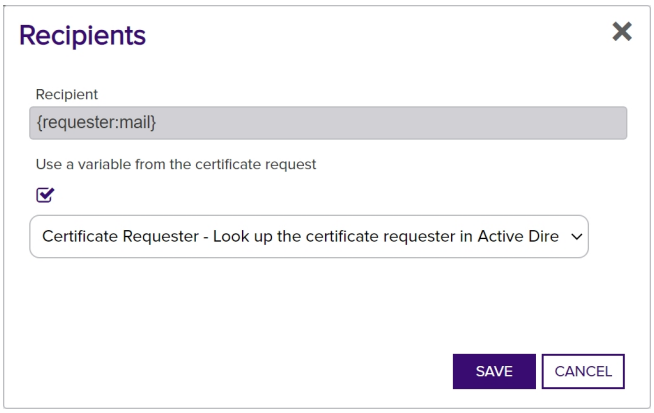


Figure 139: Denied Certificate Request Alerts Recipients

10. Click **Save** to save your denied certificate request alert.

Copying a Denied Certificate Request Alert

You may use the copy operation to create multiple similar alerts—for example, one to the requester of the certificate and another with a different message to the application owner for whom it was intended.

1. In the Management Portal, browse to *Alerts > Denied Request*.
2. On the Denied Certificate Requests Alerts page, highlight the row in the denied certificate request alerts grid and click **Copy** at the top of the grid, or from the right click menu.
3. The Denied Certificate Request Alert Settings dialog will pop-up with the details from the selected alert. The display name field will have - *Copy* tagged to the end of it to indicate it is a new alert. You may modify the alert as needed and click **Save** to add the new alert, or **Cancel** to cancel the operation.

Deleting a Denied Certificate Request Alert

1. In the Management Portal, browse to *Alerts > Denied Request*.
2. On the Denied Certificate Requests Alerts page, highlight the row in the denied certificate request alerts grid and click **Delete** at the top of the grid, or from the right click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Refer to the following table for a complete list of the substitutable special text that can be used to customize alert messages.

Table 7: Substitutable Special Text for Denied Certificate Request Alerts

Variable	Name	Description
{cmnt}	Denial	Comments provided by the administrator responsible

Variable	Name	Description
	Comments	for approving or denying the certificate request at the time the request was denied
{rcn}	Requested Common Name	Common name contained in the certificate request
{rdn}	Requested Distinguished Name	Distinguished name contained in the certificate request
{requester:mail}	Requester's Email	Email address retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:givenname}	Requester's First Name	First name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:sn}	Requester's Last Name	Last name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
{requester:displayname}	Requester's Display Name	Display name retrieved from Active Directory of the user account that requested the certificate from the CA, if present **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.

Variable	Name	Description
{careqid}	Issuing CA / Request ID	A string containing the Issuing CA name and the certificate's Request ID from the CA
{san}	Subject Alternative Name	Subject alternative name(s) contained in the certificate request
{subdate}	Submission Date	Date the certificate request was submitted
{template}	Template Name	Name of the certificate template used to create the certificate request
{templateshortname}	Template Short Name	Short name (often the name with no spaces) of the certificate template used to create the certificate request
{metadata:Email-Contact}	Email-Contact	Example of a custom metadata field
{requester:field}	String Value from AD	Locates the object in Active Directory identified by the user or computer account that requested the certificate from the CA, and substitutes the contents of the attribute named by *field*. For example, for users: • {requester:department} • {requester:sAMAccountName} For computers: • {requester:operatingSystem} • {requester:location} This substitutable special text field is partially user defined—you pick the field out of AD to include—and is therefore not available in the *Insert special text* drop-down; it needs to be typed manually. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.

2.1.6.5 Key Rotation Alerts

Key rotation alerts are used to send email notifications to SSH key users and/or administrators when a key is nearing the end of the key lifetime. The default key lifetime is 365 days, but this setting is configurable (see Application Settings: SSH Tab on page 620). Key rotation alerts apply to both user keys (see My SSH Key on page 531) and service account keys (see Service Account Keys on page 542) generated within Keyfactor Command.

The alerts can be customized to provide detailed information about the keys along with, for example, instructions to users on how to enroll for a replacement key.

Key Rotation Alert Operations

Key Rotation alert operations include: creating, editing or deleting a key rotation alert, configuring an alert schedule, copying alerts to create similar alerts for different recipients or collections, and testing alerts.

Adding or Modifying a Key Rotation Alert

1. In the Management Portal, browse to *Alerts > Key Rotation.*
2. On the Key Rotation Alerts page, click **Add** from the top menu to create a new alert, or **Edit** ,from either the top or right click menu, to modify an existing one.
3. In the Key Rotation Alert Settings dialog, select a **Timeframe** for the alert by choosing the number of days, weeks, or months to define the alert period.

   Note: When the alert is stored in the database, weeks are converted to 7 days and months are converted to 30 days.

4. In the Key Rotation Alert Settings dialog, enter a **Display Name** for the alert. This name appears in the list of key rotation alerts in the Management Portal.

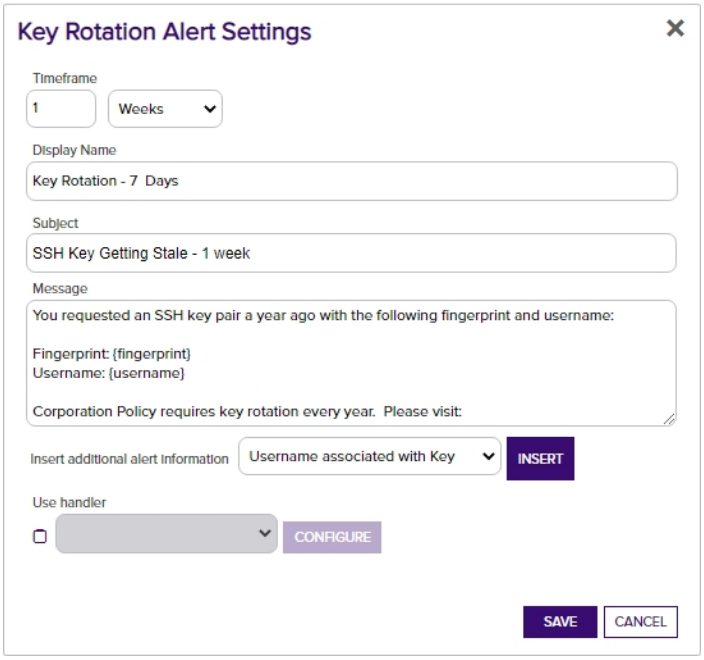


Figure 140: Key Rotation Alerts Recipients

5. In the **Subject** field, enter a subject line for the email message that will be delivered when the alert is triggered. You can use substitutable special text in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the key record at processing time. For example, you can enter {fingerprint} in the alert definition and each alert generated at processing time will contain the specific fingerprint of the given key instead of the variable {fingerprint}. To add substitutable special text to the subject line, type the special text variable enclosed in curly braces (e.g. {fingerprint}).

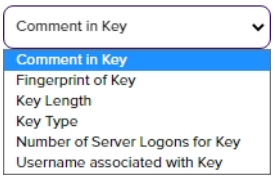


Figure 141: Substitutable Special Text for Key Rotation Alerts

6. In the **Message** box, enter the body of the email message that will be delivered when the alert is triggered. You can use the **Insert special text** dropdown below the message window to add substitutable special text to the message. Place your cursor where you would like the text to appear, select the appropriate variable from the dropdown, and click **Insert**. Alternately, you can type the special text variable enclosed in curly braces directly. If desired, you can format the message body using HTML. For example, you could place the key detail information into a table by replacing this text:

   Fingerprint: {fingerprint}

   Username: {username}

   Comment: {comment}

   With this HTML code:

   <table>

   <tr><td>Fingerprint:</td><td>{fingerprint}</td></tr>

   <tr><td>Username:</td><td>{username}</td></tr>

   <tr><td>Comment:</td><td>{comment}</td></tr>

   </table>

7. Check the **Use handler** box if you would like the alert to trigger an event handler at processing time, select the appropriate handler in the dropdown, and click the **Configure** button to configure the event handler. See Event Handler Registration on page 739 for more information on using event handlers, and Adding PowerShell Handlers to Alerts on page 223 for more information about using PowerShell Handlers.
8. Click **Save** to save your key rotation alert.

Copying an Existing Key Rotation Alert

You may use the copy operation to create multiple similar alerts—for example, one for a warning a month in advance of the stale date of keys and another shortly before the keys become stale.

1. In the Management Portal, browse to *Alerts > Key Rotation.*
2. On the Key Rotation Alerts page, highlight the row in the alerts grid and click **Copy** at the top of the grid, or from the right click menu.
3. The Key Rotation Alert Settings dialog will pop-up with the details from the selected alert. The display name field will have - *Copy* tagged to the end of it to indicate it is a new alert. You may modify the alert as needed and click **Save** to add the new alert, or **Cancel** to cancel the operation.

Deleting a Key Rotation Alert

1. In the Management Portal, browse to *Alerts > Key Rotation.*
2. On the Key Rotation Alerts page, highlight the row in the alerts grid and click **Delete** at the top of the grid, or from the right click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Configuring a Key Rotation Alert Schedule

After adding your desired key rotation alerts, you may configure a schedule to send the alerts.

1. In the Management Portal, browse to *Alerts > Key Rotation.*
2. On the Key Rotation Alerts page, click the **Configure** button at the top of the Key Rotation Alerts page to configure an alert execution schedule. This defines the frequency with which key rotation alerts are sent. This type of alert is scheduled for daily delivery at a specified time.

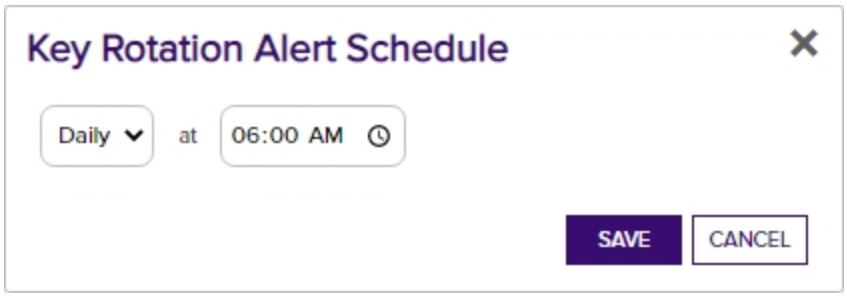


Figure 142: Key Rotation Alert Schedule

Testing Key Rotation Alerts

Once the alerts are configured, you may run a test of all or selected alerts to see if they are configured correctly.

1. In the Management Portal, browse to *Alerts > Key Rotation*.
2. On the Key Rotation Alerts page, either highlight one row in the expiration alert grid and click the **Test** button at the top of the grid or click the **Test All** button at the top of the grid to test all the alerts.
3. In the Key Rotation Alert Viewer dialog in the Alert Parameters section, select a **Start Date** and **End Date** for testing. You can use this option to simulate running the alerts a month from now instead of today, for example, or put in a broad date range to be sure you pick up some expiring certificates for testing purposes.
4. In the Key Rotation Alert Viewer dialog in the Alert Parameters section, click the toggle button for **Send Alerts** if you would like to deliver email messages as part of the test.
5. Click the **Generate** button to begin generating alerts. Depending on the number of keys to process, this may take a few seconds.
6. In the Key Rotation Alert Viewer dialog in the Alert Data and Alert Message sections, you can review the keys found to confirm that the expected keys are appearing and that the substitutable special text is being replaced as expected. Scroll through the alerts using the **First**, **Previous**, **Next** and **Last** buttons at the bottom of the dialog. The number of alerts generated will display between the navigation buttons.

Tip: Alerts are generated when an SSH key is approaching or has reached its stale date as defined by the timeframe configured in the alert and the SSH key lifetime (the *Key Lifetime (days)* application setting).
By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Key Rotation Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602 in the *Keyfactor Command Reference Guide*). If more than 100 alerts are generated, no email messages will be sent and you'll have the opportunity to view the first 100 alerts generated.

If you're using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message written) when the test is run. This is true whether or not you click the *Send Alerts* toggle.

Note: HTML does not render in the alert viewer.

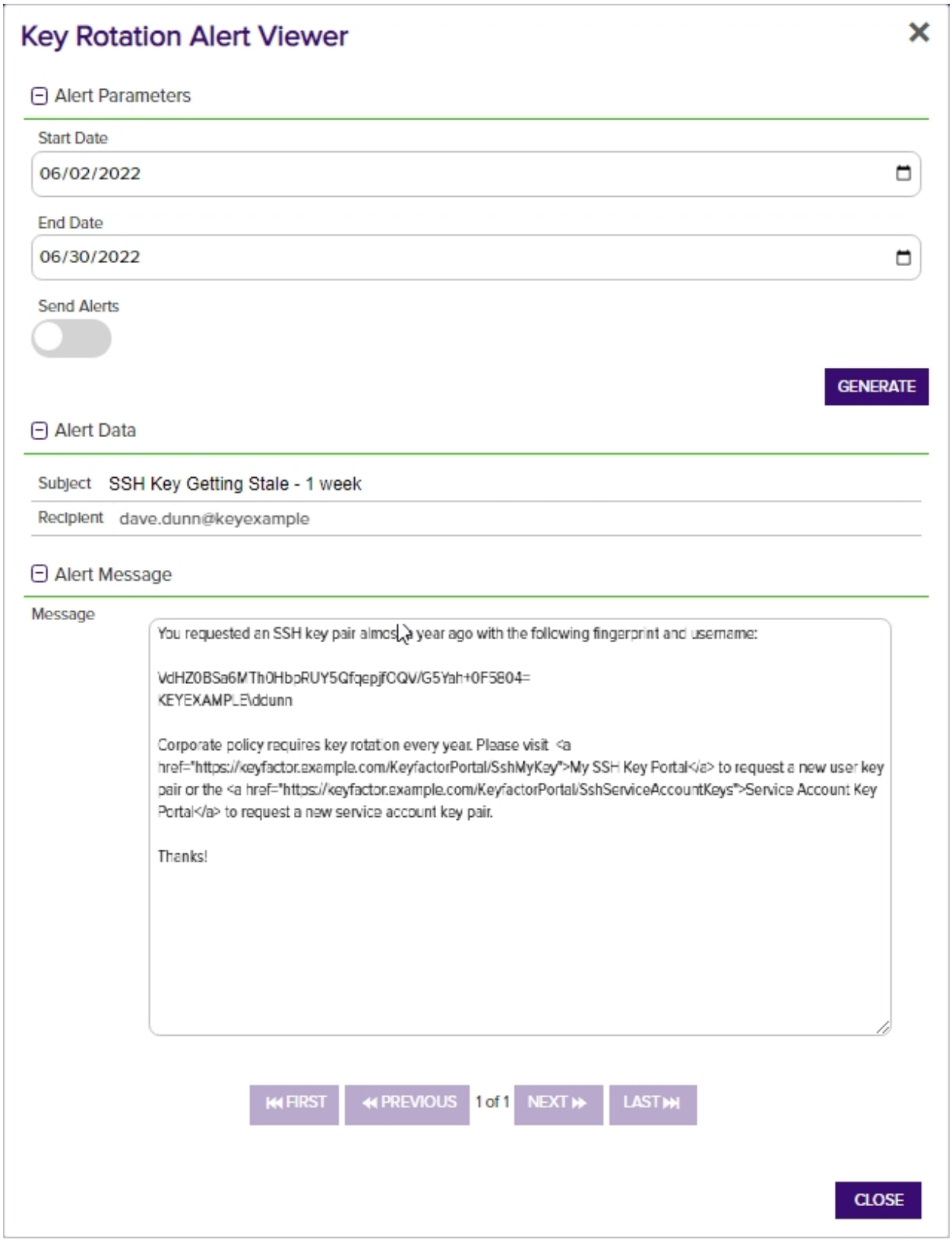

Key Rotation Alert Viewer
Alert Parameters
Start Date
06/02/2022
End Date
06/30/2022
Send Alerts
GENERATE
Alert Data
Subject SSH Key Getting Stale - 1 week
Recipient dave.dunn@keyexample
Alert Message
Message
You requested an SSH key pair almost a year ago with the following fingerprint and username:
VdHZ0BSa6MTh0HbpRUY5QfqepjfCQV/G5Yah+0F5804=
KEYEXAMPLE\ddunn
Corporate policy requires key rotation every year. Please visit My SSH Key Portal to request a new user key pair or the Service Account Key Portal to request a new service account key pair.
Thanks!
FIRST
PREVIOUS
1 of 1
NEXT
LAST
CLOSE

Figure 143: Key Rotation Alert Viewer

Refer to the following table for a complete list of the substitutable special text that can be used to customize alert messages.

Table 8: Substitutable Special Text for Key Rotation Alerts

Variable	Name	Description
{comment}	Comment in Key	The user-defined descriptive comment, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks.
{fingerprint}	Fingerprint of Key	The fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
{keylength}	Key Length	The key length for the key. The key length depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
{keytype}	Key Type	A number of cryptographic algorithms can be used to generate SSH keys. Keyfactor Command supports RSA, Ed25519, and ECDSA. RSA keys are more universally supported, and this is the default key type when generating a new key.
{serverlogons}	Number of Server Logons for Key	The number of Linux logons associated with the key, if any, granting the holder of the private key pair logon access on the server where the Linux logon resides.
{username}	Username associated with Key	The username of the user or service account associated with the key. For a user, the username is in the form of an Active Directory account (e.g. DOMAIN\username). For a service account, the username is made up of the username and client hostname entered when the service account key was created (e.g. myapp@appsrvr75).

2.1.6.6 Revocation Monitoring

Certificate revocation list (CRL) and online certificate status protocol (OCSP) locations are configured in the Revocation Monitoring section of the Management Portal to allow for email notifications when CRLs are near or at expiration, and for display on the Revocation Monitoring dashboard (see Dashboard: Revocation Monitoring on page 16). When revocation notifications are sent via email, matching events are written to the Windows event log on the Keyfactor Command server. The alert time-frame is calculated based on the date that the CRL expires, rather than the Next

Publish date. This allows for users to define their own alerts and log entries (thus determining their own definition of *stale*).

CRL monitoring and notification provides information on:

- The status of the CRL endpoint's responsiveness (e.g. is the file missing or the web site unreachable).
- Warning of upcoming expiration for a CRL.
- Notification of expired CRLs.

OCSP monitoring and notification provides only information on whether or not the OCSP endpoint is responsive. Expiration is not relevant for OSCP.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Revocation Monitoring Location Operations

From the Revocation Monitoring page on the Keyfactor Command Management Portal you can view and edit existing location endpoints, add new locations, delete an endpoint, test revocation monitoring location alert email notifications, and monitor location endpoint responsiveness.

Adding or Modifying a Revocation Monitoring Location

1. In the Management Portal, browse to *Alerts* > *Revocation Monitoring*.

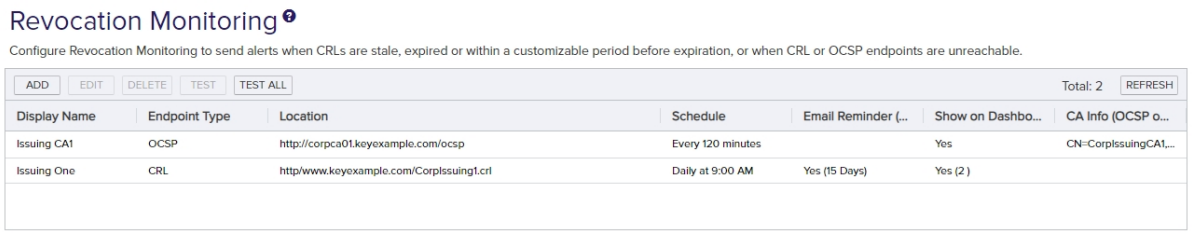


Figure 144: Revocation Monitoring Grid

2. On the Revocation Monitoring page, click **Add** to create a new monitoring location, or **Edit** to modify an existing one, and then populate the *Revocation Endpoint Settings* dialog appropriately for the type of revocation endpoint using the information below:

For a CRL location:

a. In the Revocation Endpoint Settings dialog, type a **Display Name** for the CRL location. This name appears on the Revocation Monitoring grid and on the Management Portal dashboard.

b. Select CRL in the **Endpoint Type** dropdown.

c. In the **Location** field, type a URL for the CRL location. This can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location.

> ⚠ **Important:** Because a "+" (plus sign) in a URL can represent either a space or a "+" Keyfactor Command has chosen to read "+" as a space. For CRL URLs that require a "+" (plus sign), rather than a space, replace plus signs in your CRL's URL with "`%2B`". Only replace the plus signs you don't wish to be treated as a space.

d. In the **Email Reminder** section of the page, check the **Warn** box and set the number of days ahead of expiration that email reminders should begin to be sent.

e. In the **Show on Dashboard** section of the page, check the **Warn** box and set the number of weeks, days or hours ahead of expiration for warning flags to begin appearing on the Management Portal dashboard (see Dashboard: Revocation Monitoring on page 16).

f. In the **Monitoring Execution Schedule** section of the page, configure a monitoring execution schedule. This defines the frequency with which locations are checked and alerts sent. You can choose to schedule the alert for this location either for daily delivery at a specified time or at intervals of anywhere from every 1 minute to every 12 hours. A daily schedule is the most common configuration. Schedules are configured separately for each endpoint.

g. In the **Recipients** section of the page, add email addresses of the users and/or groups who should receive email notifications when CRLs are approaching expiration or are unreachable. Recipient lists are configured separately for each endpoint.

   Keyfactor Command sends SMS (text) messages by leveraging the email to text gateways that many major mobile carriers provide. Check with your carrier for specific instructions. Keyfactor has tested that AT&T can be addressed using 10-digit-number@txt.att.net (e.g. 4155551212@txt.att.net) and Verizon can be addressed using 10-digit-number-@vtext.com (e.g. 2125551212@vtext.com). T-Mobile can be addressed using 10-digit-number@tmomail.net (e.g. 2065551212@tmomail.net), but functionality can be spotty. Reliability of alerting via this method depends on the reliability of the carrier's gateways.

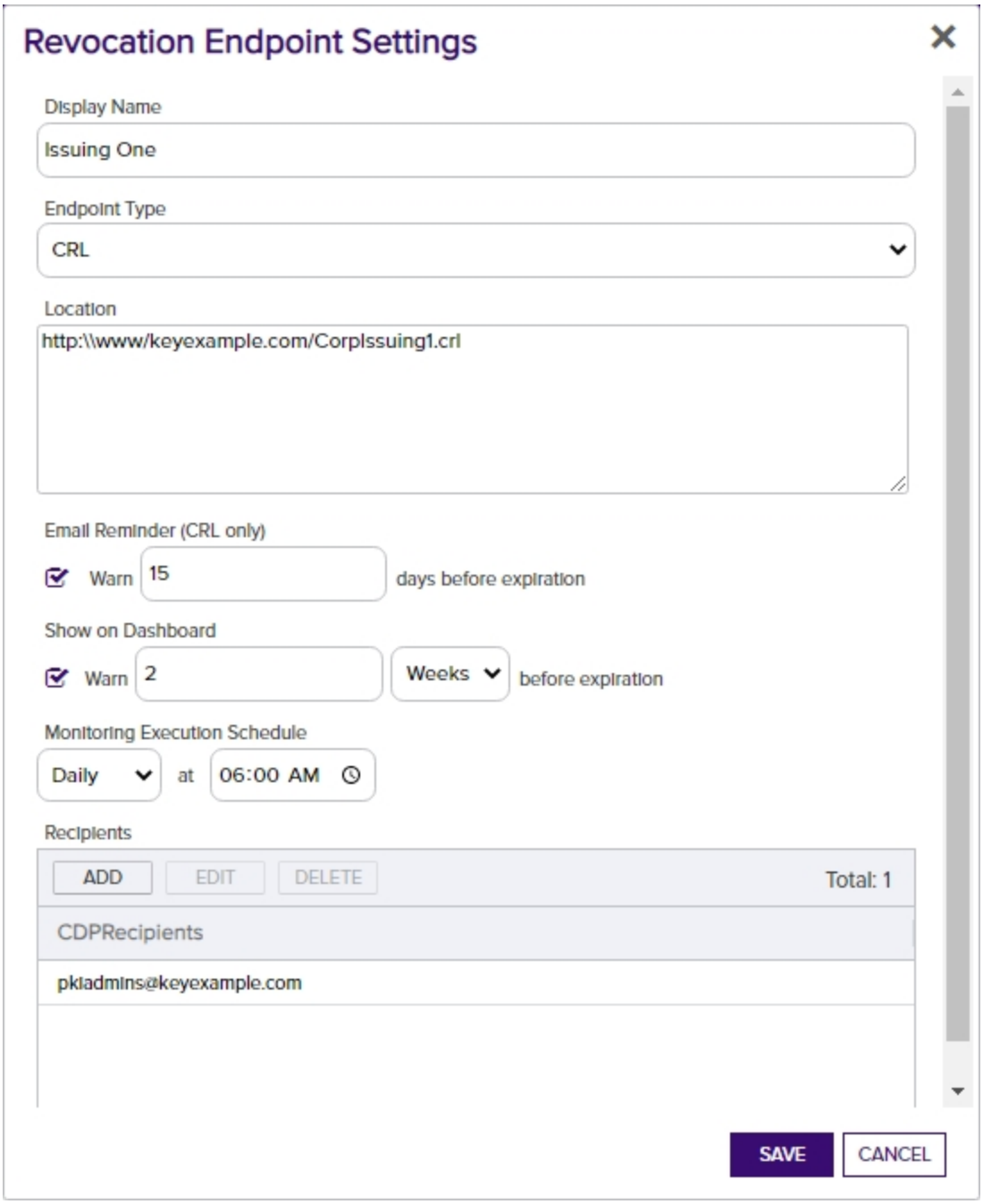


Figure 145: CRL Monitoring Details

For an OCSP location:

a. In the Revocation Endpoint Settings dialog, type a **Display Name** for the OCSP location. This name appears on the Revocation Monitoring grid and on the Management Portal dashboard.

b. Select OCSP in the **Endpoint Type** dropdown.

c. Keyfactor Command offers two options to retrieve endpoint information for OCSP:

- Resolve it based on a certificate authority defined in Keyfactor Command (see Adding or Modifying a CA Record on page 354). This option is only available for Microsoft CAs in the forest in which Keyfactor Command is installed or EJBCA CAs installed on the same network as the Keyfactor Command server. When you use this option, a request is sent for information from the Keyfactor Command server to the CA. For Microsoft CAs, this a DCOM request. For EJBCA CAs, this is a REST request.
- Import it from a certificate issued by the certificate authority to be monitored. This can be any certificate issued by the CA and containing the OCSP information. The certificate needs to be a base-64 encoded PEM file (.cer/.crt).

In the **CA Info** field, select the **CMS** radio button to automatically retrieve the CA certificate information from Keyfactor Command or select the **File** radio button to upload a file with the CA certificate information.

- If you select CMS, pick the desired CA from the CA dropdown and then click the **Resolve** button to retrieve the certificate authority information.
- If you select File, click the **Upload** button, browse to locate the file containing a certificate issued by the desired CA and open it.

  With either method of retrieving the information, you should see the full certificate authority name and authority key ID populate below the CA dropdown. The serial number field will populate for uploaded files.

d. In the **Location** section of the page, enter the full URL to the OCSP responder servicing this certificate authority's CRL.

e. In the **Show on Dashboard** section of the page, check the box to include this OCSP location on the Management Portal dashboard (see Dashboard: Revocation Monitoring on page 16).

f. In the **Monitoring Execution Schedule** section of the page, configure a monitoring execution schedule. This defines the frequency with which locations are checked and alerts sent. You can choose to schedule the alert for this location either for daily delivery at a specified time or at intervals of anywhere from every 1 minute to every 12 hours. A daily

schedule is the most common configuration. Schedules are configured separately for each endpoint.

g. In the **Recipients** section of the page, add email addresses of the users and/or groups who should receive email notifications when OCSP endpoints are unreachable. Recipient lists are configured separately for each endpoint.

   Keyfactor Command sends SMS (text) messages by leveraging the email to text gateways that many major mobile carriers provide. Check with your carrier for specific instructions. Keyfactor has tested that AT&T can be addressed using 10-digit-number@txt.att.net (e.g. 4155551212@txt.att.net) and Verizon can be addressed using 10-digit-number-@vtext.com (e.g. 2125551212@vtext.com). T-Mobile can be addressed using 10-digit-number@tmomail.net (e.g. 2065551212@tmomail.net), but functionality can be spotty. Reliability of alerting via this method depends on the reliability of the carrier's gateways.

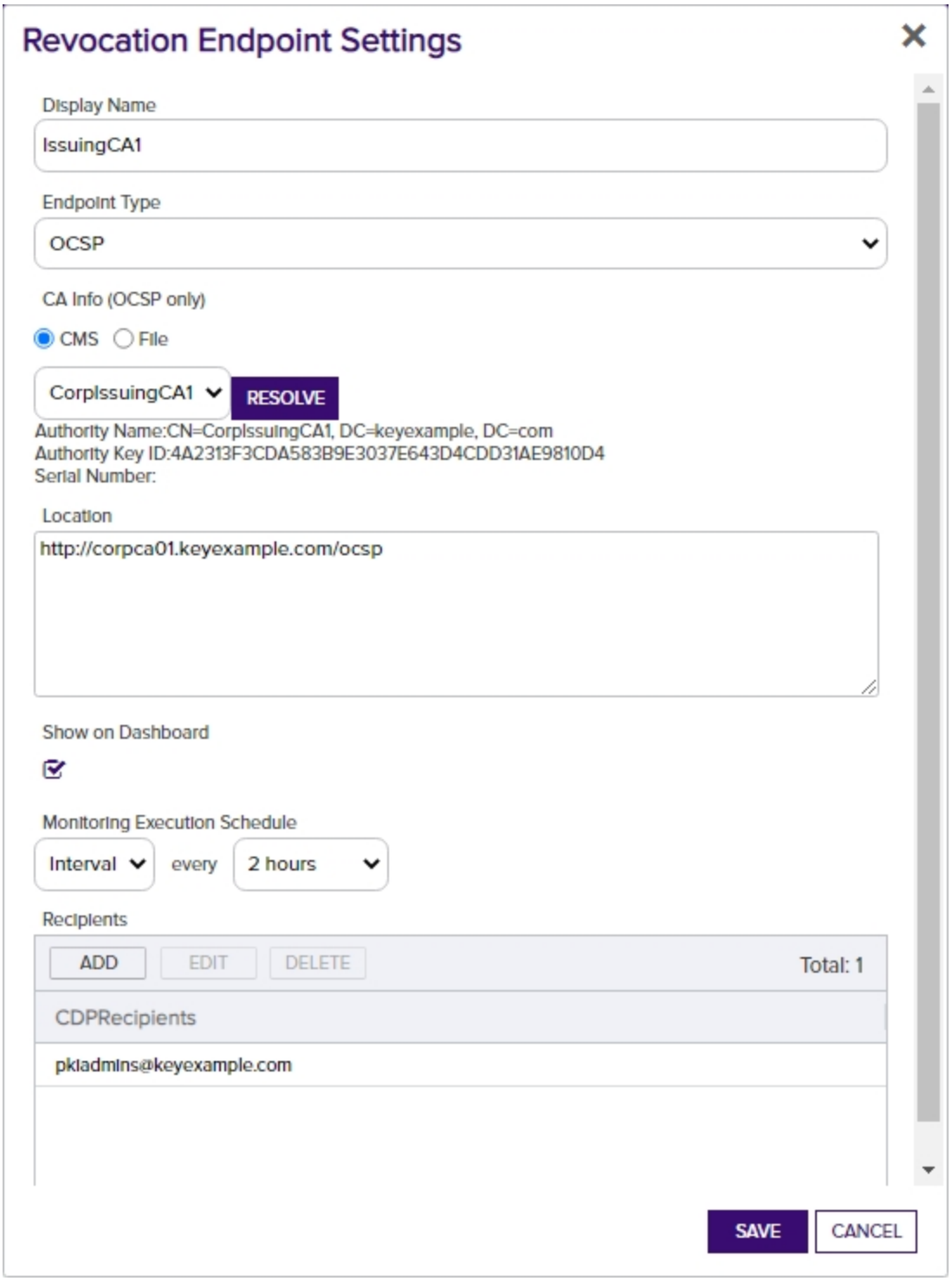

Revocation Endpoint Settings
Display Name
IssuingCA1
Endpoint Type
OCSP
CA Info (OCSP only)
CMS
File
CorpIssuingCA1
RESOLVE
Authority Name:CN=CorpIssuingCA1, DC=keyexample, DC=com
Authority Key ID:4A2313F3CDA583B9E3037E643D4CDD31AE9810D4
Serial Number:
Location
http://corpca01.keyexample.com/ocsp
Show on Dashboard
Monitoring Execution Schedule
Interval
every
2 hours
Recipients
ADD
EDIT
DELETE
Total: 1
CDPRecipients
pkiadmins@keyexample.com
SAVE
CANCEL

Figure 146: OCSP Monitoring Details

3. Click **Save** to save the endpoint location, or the changes. Click **Cancel** to cancel.

Deleting a Revocation Monitoring Location

1. In the Management Portal, browse to *Alerts > Revocation Monitoring*.
2. On the Revocation Monitoring page, highlight the row in the grid and click **Delete** at the top of the grid, or from the right click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Testing Revocation Alerts

1. In the Management Portal, browse to *Alerts > Revocation Monitoring*.
2. On the Revocation Monitoring page, click the **Test All** button at the top of the grid, or select a specific location from the grid and click **Test** from the top of the grid or the right click menu.
3. In the Revocation Monitoring Test dialog in the Alert Parameters section, select an **End Date** for testing. You can use this option to simulate running the alerts a month from now instead of today, for example, or put in a date far in the future to be sure you pick up some expiring CRLs for testing purposes.
4. In the Revocation Monitoring Test dialog in the Alert Parameters section, click the toggle button for **Send Alerts** if you would like to deliver email messages as part of the test.
5. Click the **Generate** button to begin generating alerts. Depending on the number of endpoints to process, this may take a few seconds.
6. In the Revocation Monitoring Test dialog in the Alert Data and Alert Message sections, you can review the alerts to confirm that the expected CRLs and OCSP endpoints are appearing. Scroll through the alerts using the **First**, **Previous**, **Next** and **Last** buttons at the bottom of the dialog. The number of alerts generated will display between the navigation buttons.

Tip: Alerts are generated when a CRL is expired or in the warning period as defined by the number of days configured in the *Email Reminder* setting. For example, if you had a CRL that expired on June 30 and configured the email reminder period to 15 days before expiration, the warning status would begin for that CRL on June 15 and CRL alerts would be generated. A warning will also appear for any CRL or OCSP locations that produced an error or couldn't be resolved.
When alerts are tested or sent on a schedule, corresponding message are also written to the system event log on the server where the Keyfactor Command service runs. For testing, this is true whether or not you click the *Send Alerts* toggle. Information is logged to the event log for both locations that are in a good state (e.g. CRL resolves and is not in a warning or expired state or response from OCSP) and locations that are in an error state (e.g. CRL resolves but is in the warning period or expired, CRL is expired, CRL or OCSP location does not resolve).

For specific Windows event ID information, see Keyfactor Command Windows Event IDs on page 806.

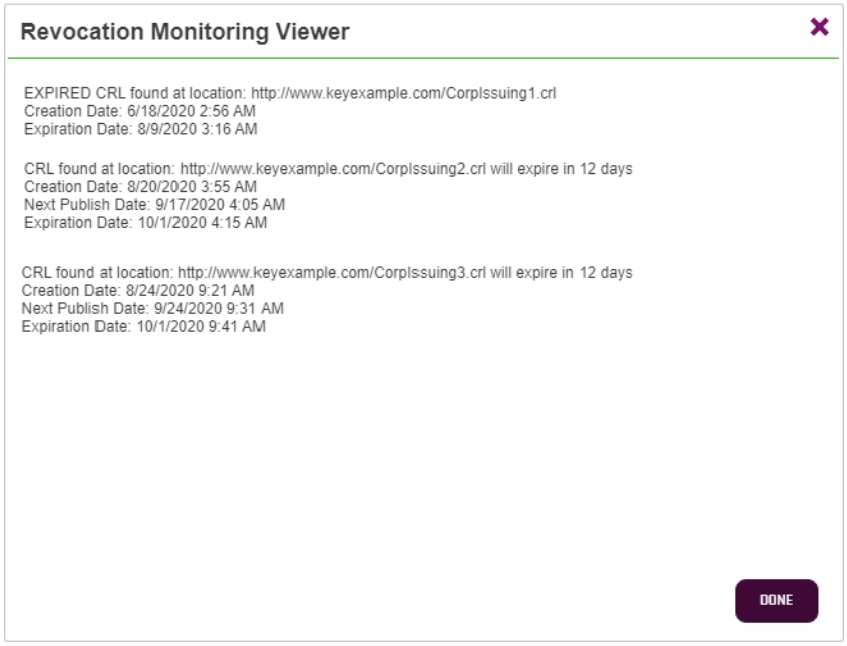


Figure 147: Test Revocation Monitoring

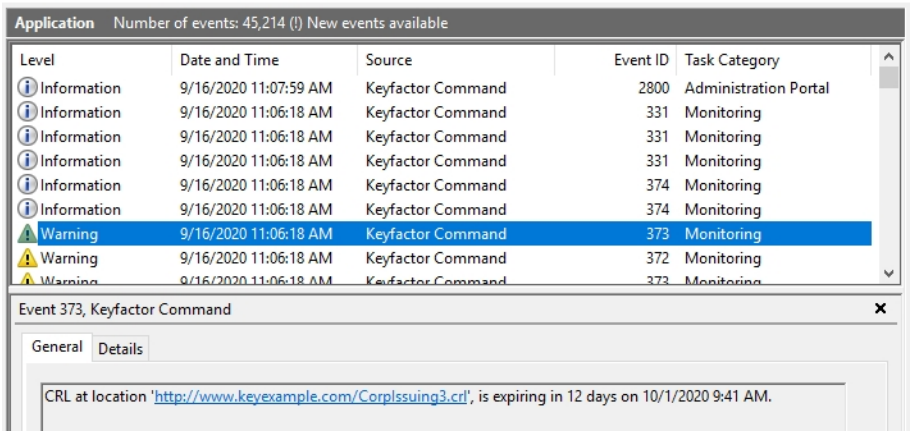


Figure 148: Revocation Monitoring Event Log Messages

2.1.6.7 Using Event Handlers

A given expiration, pending, issued or denied alert can have only one event handler action associated with it. For example, an alert can run one PowerShell script but not also a second PowerShell script or also an event logging task. Alerts configured with a PowerShell or renewal event handler can also send out email messages. However, be aware that your PowerShell script will run once for

every certificate and every email recipient, so if your alert has three email recipients, your script will run three times for each certificate. If this is not the desired behavior, you can set up separate alerts for email messages and your PowerShell script. Alerts configured with an event logger event handler will log events to the event log instead of sending email messages. If you want to both log to the event log and send email messages for a given alert configuration, you need to set up two separate alerts.

Tip: PowerShell handlers will run in different security contexts depending on how they are triggered. If they are triggered by the Management Portal/Keyfactor API they will run in the context of the Keyfactor API application pool account. If they are triggered by a task scheduled in the Keyfactor Command Management Portal, they will run in the context of the Keyfactor Command Service account. Keep this in mind if your configuration of the PowerShell script is going to use Windows Authentication to reach back into Keyfactor Command,or elsewhere.

PowerShell Scripts

PowerShell scripts used in alert event handlers and workflows are stored in the Keyfactor Command database and need to first be imported into the database using the *POST /Extensions/Scripts* API endpoint (see POST Extensions Scripts on page 1709) before they will be available for use in alerts and workflows. Scripts can only be managed through the Keyfactor API.

Use with Workflow

Within workflow step definitions, PowerShell can be used with the step types **Set Variable Data** and **Use Custom PowerShell**. Only the **Use Custom PowerShell** step uses the /Extensions/Scripts API endpoints. Once imported, scripts will be available for selection from a dropdown in the workflow workspace on the step dialogue when creating a step of this type or with the workflow definition API endpoints. The **Set Variable Data** step allows PowerShell commands to be added to the step directly, thus are not stored in the database as scripts, but only with the workflow step. Many examples of using scripts with workflow can be found in Workflow Definitions Configuration Parameters on page 251.

Use with Event Handlers

Event handler scripts are available for use in multiple Keyfactor Command areas, including:

- Expiration Alerts on page 167
  Create email notifications that alert administrators and/or end users when certificates are coming up for expiration.
- Pending Certificate Request Alerts on page 178
  Create email notifications that alert PKI administrators when a new pending certificate request is made.
- Issued Certificate Request Alerts on page 188
  Create email notifications that alert a certificate requester when a certificate he or she requested has been issued.

- Denied Certificate Request Alerts on page 197
  Create email notifications that alert a certificate requester when a certificate he or she requested has been denied.
- Key Rotation Alerts on page 203
  Create email notifications that alert end users and PKI administrators when an SSH key is nearing the end of its lifetime.

Event handler scripts used in any of these areas need to be imported into Keyfactor Command before they will be available for use in the alerts.

To create a PowerShell script that works with the event handlers, there are just a few things to keep in mind:

- You need to declare the $context hashtable at the start of the script with this line:

  ```
  [hashtable]$context
  ```

- Parameters you want to use in your script are referenced using the $context syntax as follows (where MyName is the name you gave to the parameter in the event handler configuration or the name of the built-in parameter from Table 9: PowerShell Event Handler Special Fields):

  ```
  $context["MyName"]
  ```

Table 9: PowerShell Event Handler Special Fields

Name	Alert Type	Description
SendEmail	All	If true, email messages are sent in addition to processing of the PowerShell script.
Subject	All	The full subject line of the alert.
Message	All	The full message body of the alert.
Recipient	All	The recipient of the alert. Alerts configured with more than one recipient will execute the PowerShell script multiple times—once for each recipient and each certificate or request.
Certificate	Expiration Only	For internal Keyfactor use only.
First Recipient	Expiration Only	If true and the alert has multiple recipients configured, this output is for the first recipient for the given certificate. Subsequent output for the same certificate and different recipients will show false for this value.

Tip: A sample PowerShell script is installed with Keyfactor Command in the ExtensionLibrary directory, located by default at *C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\net6.0\SampleEventHandler.ps1*.

Below is a simple script that takes as inputs all the parameters you defined in your event handler configuration as well as the built-in parameters and outputs them to a file along with a comment and the date, with configuration to skip output of a defined list of the built-in parameters:

```
# This is a sample script that can be set up as a Keyfactor Command event handler. The script will
output
# data passed to the handler to a text file. This script will be called for the combination of each
# certificate involved in the corresponding event and each configured email recipient.

# In order to communicate with the extension script, the Keyfactor Command event handler framework
injects
# a hashtable into the PowerShell runspace. This hashtable will include the fields configured by the
# administrator when setting up the handler as well as some built-in system fields used for commu-
nication
# with the handler.

# The following fields are provided for communication with the handler:
#   Subject -        Email subject line that will be sent if the alert has the email subject
configured
#   Message -        Email body that will be sent if the alert has the email message body configured
#   Recipient -      Email address where the alert will be sent if the alert has this configured
#   Certificate -    For internal use only
#   SendEmail -      Boolean (true/false) indicating if Keyfactor Command is planning on sending an
email
#                    for this certificate / recipient combination
#   FirstRecipient - Boolean (true/false) indicating if this extension invocation is the first recip-
ient
#                    for a given certificate
#                    This can be used in the event it is desired to execute some logic once per certi-
ficate
#                    This field applies only to expiration alerts

[hashtable]$context

# Four of the built-in context fields can be modified and used as output fields to change how (and
if)
# Keyfactor Command will send emails related to the alert being processed:
#   Subject - If an email is produced this new value will be used to create the email subject.
#   Message - If an email is produced this new value will be used to create the email message body.
```

```
#   Recipient - If an email is produced this new value will be used as the email recipient.
#   SendEmail - This value can be used to override whether an email will be sent.
#     A value of "true" will cause an email to be sent, while "false" will cause the associated email
#     to not be sent.
#     Examples:
#       $context["Subject"] = "new subject line"
#       $context["Message"] = "new message line"
#       $context["Recipient"] = "newRecipient@keyexample.com"
#       $context["SendEmail"] = "false"

# Typically output values would be used with some form of logic. As an example, to change the recip-
ient
# of the email based on a metadata field provided to the handler, uncomment the following, provide
# appropriate values (including a metadata field that's being passed in to the handler in place of
# "SampleMetadataField"), and remove Recipient from ignoreKeys:
#
#      if ($context["SampleMetadataField"] -eq "SomeValue") {
#              $context["Recipient"] = "newRecipient@keyexample.com"
#              }

# This example will output to a file the $context values for the user configured fields and skip the
system
# supplied ones. To output the system supplied fields, remove the desired items from the $ignoreKeys
array.
$ignoreKeys = "Subject", "Message", "SendEmail", "Certificate", "FirstRecipient", "Recipient"

# Path to the output file
$outputFile = ("C:\PSScripts\Output\SampleScriptOutput" + (get-date -UFormat "%Y%m%d%H%M") + ".txt")

# Add a comment and the date at the start of each output block
Add-Content -Path $outputFile -Value "Starting Output: $(Get-Date -format G)"

# Loop through all passed in key/value keys and process
$context.GetEnumerator() | % {
       if (-not $ignoreKeys.Contains($_.key)) {
              Add-Content -Path $outputFile -Value ($_.key + ": " + $_.value)
       }
}

# Add a blank line between output blocks
Add-Content -Path $outputFile -Value ""
```

Adding PowerShell Handlers to Alerts

To add a PowerShell handler to an alert, the alert must first be created and saved. See Alerts on page 166 for more information on creating various alerts. The example below uses an expiration alert, but the process applies to all types of alerts.

Next, you must create the PowerShell script and import it into Keyfactor Command. See PowerShell Scripts on page 219 for important information regarding working with scripts.

Note: See PowerShell Scripts on page 219 for important information regarding working with scripts.

1. Select the alert to which you want to add the event handler from the respective alert grid.
2. Check the **Use handler** box and select the PowerShell event handler in the dropdown.

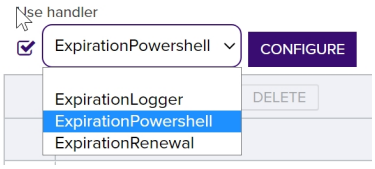


Figure 149: Use PowerShell Expiration Event Handler

Tip: If the expected event handler types do not appear, confirm that they exist and are enabled on the Event Handler Registration page (see Event Handler Registration on page 739).

3. Click the **Configure** button in the Use handler section of the page to open the Configure Event Handler dialog and then click **Add**.

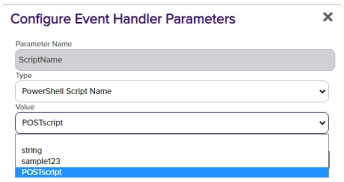


Figure 150: Expiration Alert with PowerShell Event Handler

4. In the Configure Event Handler Parameters dialog, select **PowerShell Script Name** as the parameter Type. The Parameter Name field will change to *ScriptName*. Select the desired script from the dropdown, Only scripts that are configured for the selected alert category and imported to the Keyfactor Command database will display in the dropdown. See PowerShell Scripts on page 219 for more information on script handling.
5. Click **Save** to save your first parameter.

6. If desired, you can pass one or more parameters into your PowerShell script—either fixed text (type Static Value) or substitutable special text (type Special Text). To pass in fixed text, enter a name for the parameter (e.g. MyName), select the **Static Value** radio button, and type your fixed text in the Value field. To pass in special text, enter a name for the parameter (e.g. MyOtherName), select the **Special Text** radio button, and select your desired substitutable special text field in the Value dropdown. When referring to these parameters in your PowerShell script, refer to them using a *$context* hashtable parameter passed to the script, whose keys are the names entered in the event handler configuration. See Figure 151: PowerShell Event Handler with Multiple Parameters. For example, for the parameter named "cn" in the event handler configuration, you might use this line in a PowerShell script:

   ```
   if ($context.ContainsKey("cn")) { Add-Content -Path "C:\Stuff\MyOutput.txt" -Value $context["cn"] }
   ```

   In addition to the parameters you opt to pass in the event handler configuration, there are several built-in parameters that are always passed. These can be found in Table 10: PowerShell Event Handler Special Fields. You can reference these in your PowerShell script without having to specify them in your event handler configuration.

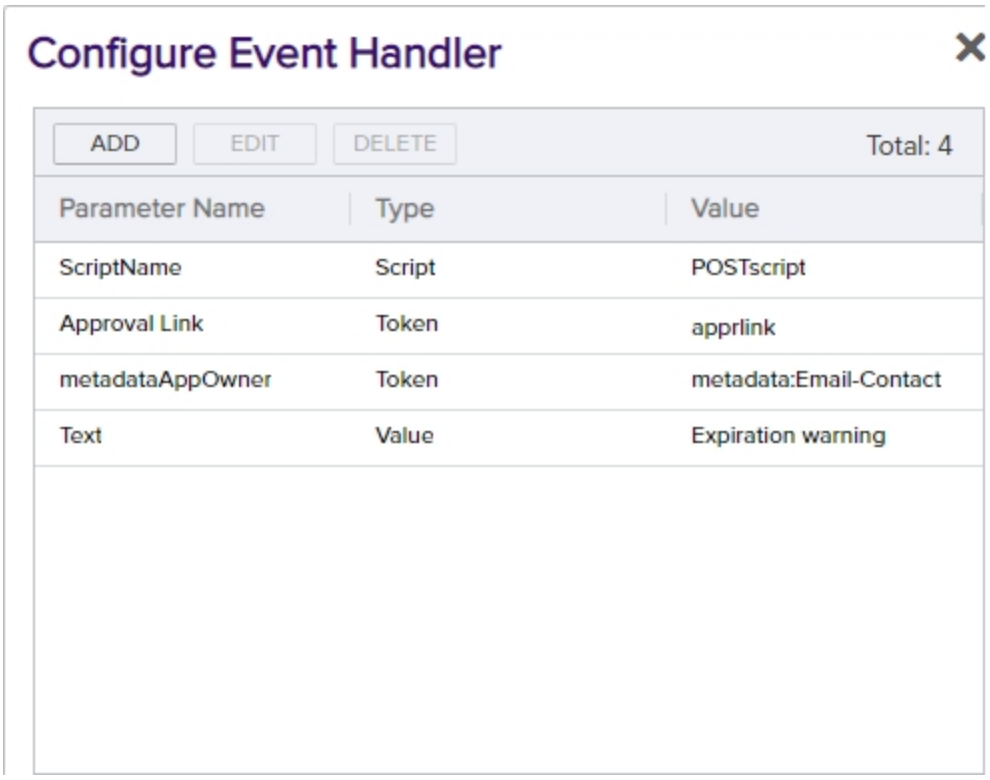


Figure 151: PowerShell Event Handler with Multiple Parameters

7. Click **Close** to return to the alert configuration and then save the alert.

Table 10: PowerShell Event Handler Special Fields

Name	Alert Type	Description
SendEmail	All	If true, email messages are sent in addition to processing of the PowerShell script.
Subject	All	The full subject line of the alert.
Message	All	The full message body of the alert.
Recipient	All	The recipient of the alert. Alerts configured with more than one recipient will execute the PowerShell script multiple times—once for each recipient and each certificate or request.
Certificate	Expiration Only	For internal Keyfactor use only.
First Recipient	Expiration Only	If true and the alert has multiple recipients configured, this output is for the first recipient for the given certificate. Subsequent output for the same certificate and different recipients will show false for this value.

Adding Logging Handlers to Alerts

To add a logging handler to an alert:

1. Edit an existing alert or create a new one. An alert cannot both send emails and write to the event log, so if you need to do both of these for the same alert configuration, you will need two separate alerts.
2. Configure the message body as you would for an email message, including substitutable special text. The text from the message body is written to the event log. Note that HTML is not supported in the message body for event logging. The contents of the *Subject* line do not appear in the event log.
3. Check the **Use handler** box and select the logger event handler in the dropdown.

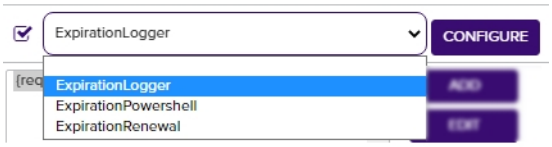


Figure 152: Expiration Alert with Event Logging Event Handler

Tip: If the expected event handler types do not appear, confirm that they exist and are enabled on the Event Handler Registration page (see Event Handler Registration on page 739).

4. Click the **Configure** button in the Use handler section of the page to open the Configure Event Handler dialog and then click **Add**.

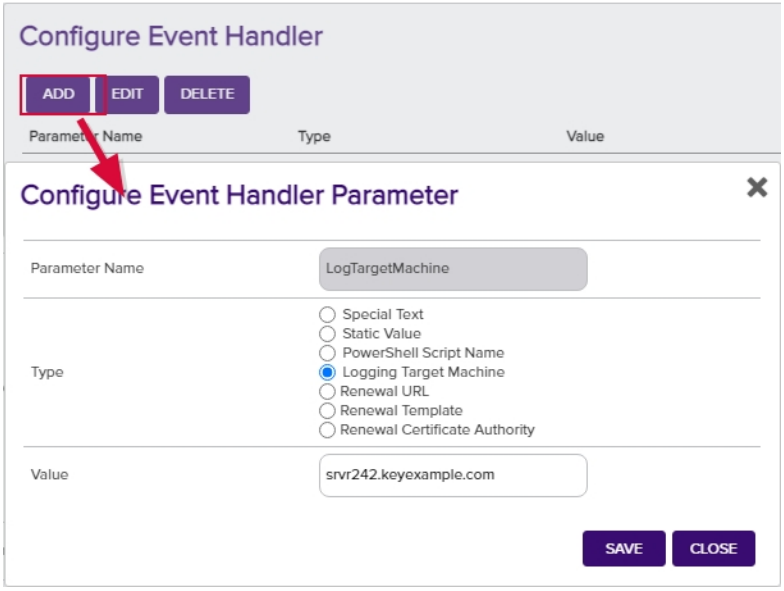


Figure 153: Expiration Alert with Logging Event Handler

5. In the Configure Event Handler Parameter dialog, select **Logging Target Machine** as the parameter Type, and enter the fully qualified domain name of the server to which you wish to send the event log message in the Value field.

   By default, the service accounts under which the Keyfactor Command application pool and Keyfactor Command service run have sufficient permissions to write to the event log on the Keyfactor Command server. If your target computer is not the Keyfactor Command server, you will need to grant appropriate permissions on that computer to one or both of these service accounts in order to write to the event log on that computer. When alerts containing event handlers are run in test most, the application pool service account is used. When alerts containing event handlers are run as a scheduled task, the Keyfactor Command service account is used. Local administrator permissions are needed initially to allow the service account to create the event log source types on the target machine. After that has been completed (on the first successful write of event logs to the server), permissions for the service account can be dialed back to “Generate security audits” or “Manage auditing and security log” in the local security policy.

If you wish to use a DNS alias for the target machine value, you may need to disable loopback checking on the Keyfactor Command server and reference the target machine. See Disable Loopback Checking on page 832.

6. Click **Save** to save and then **Close** to return to the alert configuration. No other parameters are needed (or functional) for an event logging event handler.

7. Test the alert as described in Expiration Alerts on page 167. It is not necessary to check the **Send Alerts** box during the test. Alerts are written to the Application event log.

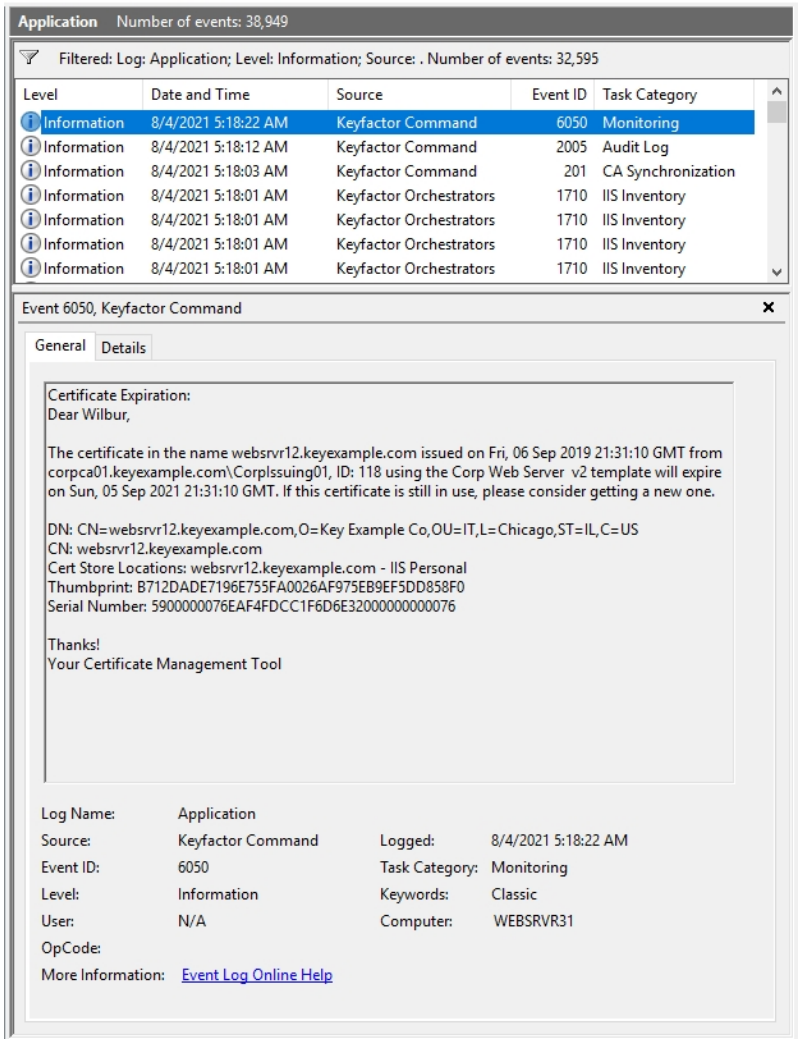


Figure 154: Expiration Alert Event Log

Adding Renewal Handlers to Expiration Alerts

Important: Renewal alerts will not function until you configure security permissions for the renewal handler as per Configure Renewal Handler Permission on page 2841 in the *Keyfactor Command Server Installation Guide*.

To add a renewal handler to an expiration alert:

1. Edit an existing expiration alert or create a new one. See Expiration Alert Operations on page 167.
2. Check the **Use handler** box and select the renewal event handler in the dropdown.

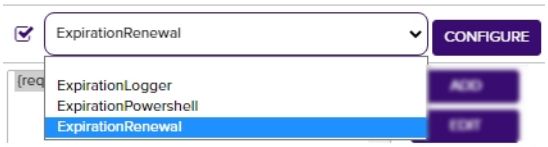


Figure 155: Use Renewal Event Handler on Expiration Alert

Tip: If the expected event handler types do not appear, confirm that they exist and are enabled on the Event Handler Registration page (see Event Handler Registration on page 739).

3. Click the **Configure** button in the Use handler section of the page to open the Configure Event Handler dialog and then click **Add**.

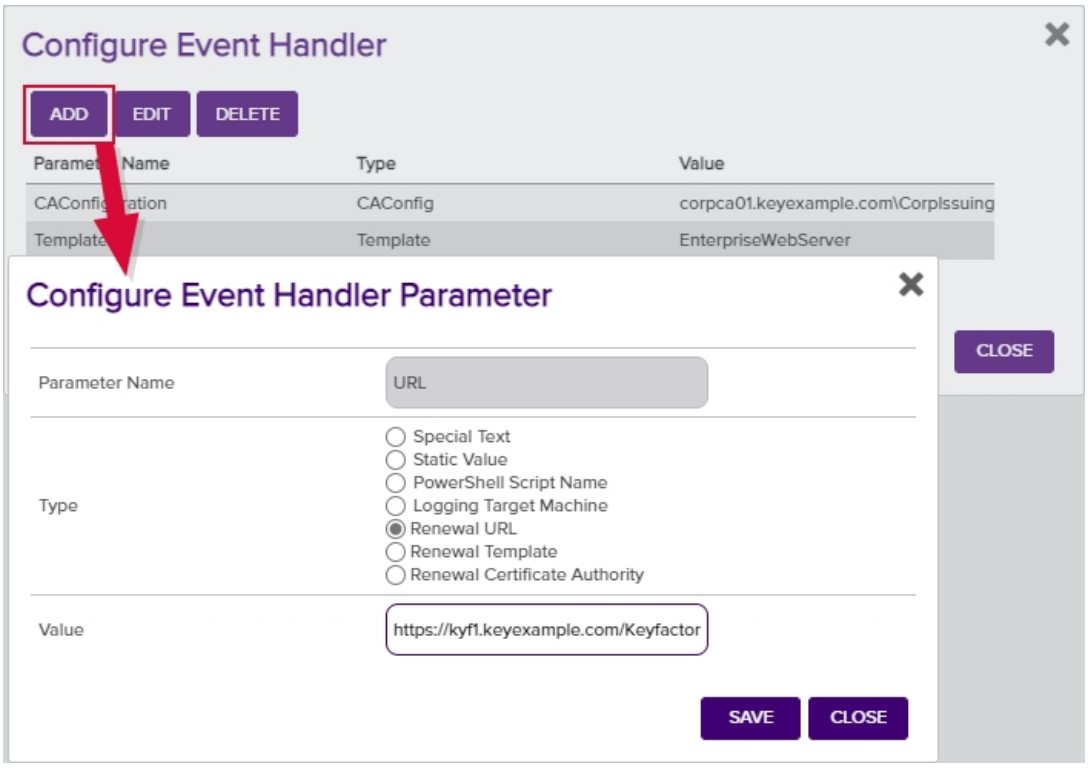


Figure 156: Expiration Alert with URL Event Handler

4. In the Configure Event Handler Parameter dialog, select **Renewal URL** as the parameter Type, and enter the URL to the Keyfactor Command server hosting the Keyfactor API component followed by /KeyfactorApi in the Value field. Click **Save** to save your first parameter.

5. If desired, you can configure a renewal template and CA for use with the renewal event handler. These settings are optional. If you don't set these, the renewal will be done using the template and CA originally used on the certificate. If you set only one of these—for example, the template—it will use the setting from the renewal event handler for that and retrieve the other—for example, the CA—from the certificate.

6. Test the alert as described in Expiration Alerts on page 167. It is not necessary to check the **Send Alerts** box during the test.

> ⚠ **Important:** Renewals **are** processed and new certificates **are issued** during expiration alert tests with associated renewal handlers.

2.1.7 Workflow

The options available in the Workflow section of the Management Portal are:

- **Workflow Definitions**

  Create workflows that manage certificate enrollments, renewals, or revocations end-to-end to require approvals, send emails, run PowerShell scripts and/or execute API requests as part of the process and workflows that are initiated by an automated task that runs periodically (every 10 minutes by default) to identify additions and removals of certificates from a specified certificate collection.

- **Workflow Instances**

  Manage initiated instances of workflows to view active, suspended (requiring approval) and completed enrollments, renewals, revocations, and additions and removals of certificates from a specified certificate collection. This page allows you to view the steps in a given instance of a workflow (which may be different from the current configuration of the workflow definition), restart failed workflow instances, and delete workflow instances.

- **My Workflows**

  Review initiated instances of workflows awaiting action by you and take action (e.g. approve or deny enrollment or revocation requests) or created by you.

2.1.7.1 Workflow Definitions

The workflow builder in Keyfactor Command allows you to easily automate event-driven tasks to manage certificate enrollments, renewals, revocations on a per template basis. It can also monitor certificate collections on a periodic basis for certificates that change membership status based on the query criteria of a specified certificate collection. The workflows can be configured with multiple steps between the start and end of the operation that offer a simple way to configure notifications, approvals, and end-to-end automation throughout the environment. This provides for operational agility in an intuitive and easy-to-configure manner.

When a user begins one of the types of actions managed with workflow in Keyfactor Command on the usual Management Portal page (e.g. PFX Enrollment) or using the Keyfactor API or a certificate collection membership change is detected by an automated task, the workflow kicks in behind the scenes and executes however many steps have been configured in the workflow definition to bring the action to the appropriate conclusion along the desired path.

See Certificate Collection Manager on page 85 for more information about creating certificate collections.

Workflow Types

The following types of workflow triggering events are supported:

- **Certificate Entered Collection**

  The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection.

For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered.

- **Certificate Left Collection**

  The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection.

  For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate.

- **Enrollment (Including Renewals)**

  The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.

- **Revocation**

  The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.

Workflow Steps

The following customizable workflow steps are supported within the workflows:

- **Send Email**

  Send an email message. This is a separate email message from those typically sent as part of a *Require Approval* step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request.

- **Set Variable Data**

  Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including:

  - ConvertFrom-Csv
  - ConvertFrom-Json
  - ConvertFrom-Markdown
  - ConvertFrom-SddlString
  - ConvertFrom-StringData
  - ConvertTo-Csv

- ConvertTo-Html
- ConvertTo-Json
- ConvertTo-Xml
- ForEach-Object
- Get-Command
- Where-Object

- **Use Custom PowerShell**

  Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See Extensions Scripts on page 1704 for adding scripts to the database.

- **Require Approval**

  Require approval for a workflow step before the step can be completed. The require approval step can require approval from just one approver or multiple approvers. The workflow will be suspended at this point until the correct number of approvals from users with the correct security roles is received or until one deny is received before continuing to the next step. As part of this step, an email message is sent indicating whether the step was approved or denied—typically to the requester. This step does not include logic to send an email initiating the approval process (letting users know something needs approval). Use a *Send Email* type step for this.

  Important: Workflows are not supported with CA delegation when they contain steps that require approval. For more information, see the CA configuration Authorization Methods Tab on page 367.

  Note: The users that you send email to initiating the approval process must be members of a security role that is allowed to submit signals (approve/deny) for the workflow in order to approve or deny the request.

  Tip: The workflow builder does not include a step to send a notification to the requester of a certificate on an enrollment once the certificate is issued by the CA (as opposed to approved in Keyfactor Command). Use the issued alerts for this (see Issued Request Alert Operations on page 189).

- **Invoke REST Request**

  Run a REST (API) request using Active Directory as an identity provider and Basic or Windows authentication. The REST request contents are embedded within the step. It does not call out to an external file.

- **Invoke REST Request with OAuth**

Run a REST (API) request using an identity provider other than Active Directory and Token authentication. The REST request contents are embedded within the step. It does not call out to an external file.

- **Update Certificate Request Subject\SANS for MSFT CAs (Enrollment Only)**

  On an enrollment (either CSR or PFX), create a resigned CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) to modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types or a custom step type. The step creates a new CSR using the same public key as the original CSR using the updated SAN and/or subject values. It signs the new CSR with the certificate provided in the step's configuration.

  For this type of step you will need an enrollment agent certificate available as a PKCS#12 (.PFX) file with included private key to import into Keyfactor Command. This can be a user certificate or a computer certificate (e.g. generated from a copy of the *Enrollment Agent* template or the *Enrollment Agent (Computer)* template) and must have a Certificate Request Agent EKU. Note that the built-in *Enrollment Agent* and *Enrollment Agent (Computer)* templates do not allow private keys to be exported by default. You will need a template that allows private key export or will need to manually override private key export to create a certificate with an exportable private key in order to create a PKCS#12 (.PFX) file.

  > **Important:** This step applies to Microsoft CAs only. If this step is added to workflow for requests directed to an EJBCA CA, it will fail on enrollment. Note that EJBCA supports submission of updated SAN or subject details as part of standard functionality.

- **Windows Enrollment Gateway - Populate from AD (Enrollment Only)**

  On an enrollment done through the Keyfactor Windows Enrollment Gateway using a client-side template configured with the *Build from this Active Directory information* option on the template, this workflow step handles formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as *Build from this Active Directory information* must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step.

  > **Important:** The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the *Keyfactor Windows Enrollment Gateway Installation and Configuration Guide*.

In addition to these customizable types of steps, there are built-in steps that you won't see unless you're using the Keyfactor API to view or edit the workflows (see Workflow Definitions on

page 2487). At the end of their respective workflow types there are an enroll step and a revoke step to initiate the actual enrollment or revocation if the workflow reaches the end without being denied or failing. These built-in steps cannot be modified or moved to a different location in the workflow. There are also NOOP steps that indicate the start and end of the workflow for housekeeping purposes.

There are two types of workflow definition:

- **Global**

  The global workflow definitions are built into the product and cannot be deleted, though they can be modified to add workflow steps, if desired. Global workflow definitions do not have a specific associated *key*—in the case of the currently available workflows, this is a certificate template—and apply to all requests of the workflow's type (e.g. enrollment) that are not otherwise handled by a custom workflow specifying a *key*.

- **Custom**

  Custom workflow definitions are any additional workflow definitions you define beyond the built-in ones. Custom workflows are associated with a specific *key* (certificate template or certificate collection) and each workflow only applies to requests made using that *key*.

Note: All certificate enrollment, renewal, and revocation requests go through workflow even if you haven't created any workflow steps or added any custom workflow definitions. In the absence of customization, the global workflow definitions are used. The addition and removal of certificates from certificate collections only go through workflow if you create custom workflows for them.

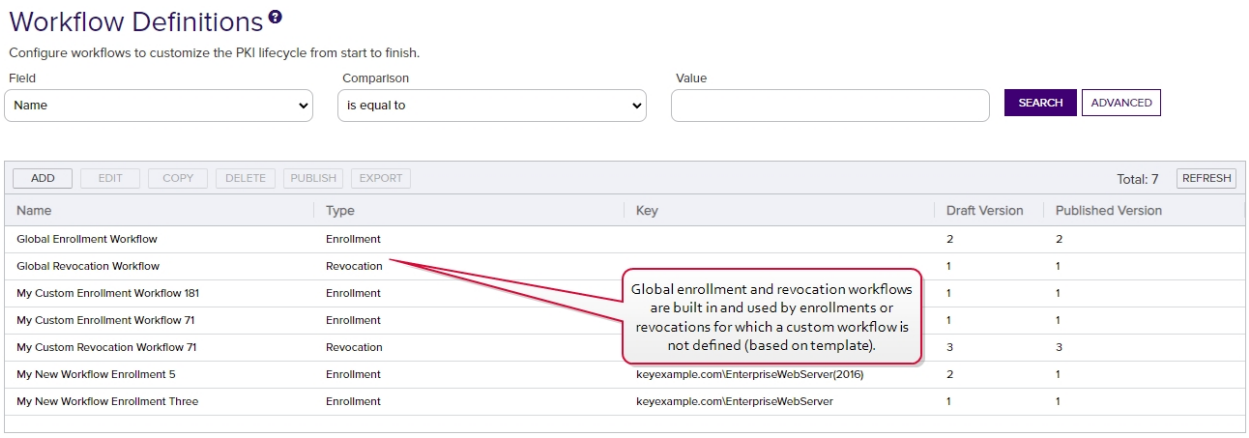


Figure 157: Workflow Definitions

When requiring approval for enrollment using workflow definitions in Keyfactor Command, templates do not need to be configured to require manager approval at the CA level in the certificate template. This is because the approval handling is fully controlled within Keyfactor Command. In fact, templates generally should not be configured to require CA manager approval when using Keyfactor Command workflow, since this would generally require approval both at the Keyfactor Command level and at the CA level.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Workflow Definition Operations

The workflow builder in Keyfactor Command is a powerful feature that allows you to manage certificate enrollments, renewals, and revocations on a per template basis, end-to-end. It can also monitor certificate collections on a periodic basis for certificates that change membership status based on the query criteria of a specified certificate collection. Out of the box, there are workflow builder steps such as requiring approvals for actions like certificate enrollment and revocation requests, sending email notifications, and running PowerShell scripts and API requests as part of the request flow.

Tip: There are two built-in workflow definitions—Global Enrollment Workflow and Global Revocation Workflow—that are used to manage enrollment and revocation requests which are not otherwise handled by custom workflows. These workflows can be configured with steps (see Adding, Copying or Modifying a Workflow Definition on page 237), but they cannot be deleted. There are no built-in workflow definitions for the addition and removal of certificates from certificate collections. These actions only go through workflow if you create custom workflows for them.

Working with the Keyfactor Command Workflow Builder Workspace

The workflow builder workspace is laid out with the workflow steps running from top to bottom in the middle (initially), the Workflow Definition dialog in a collapsible window on the right, and workspace controls at the bottom left. If you create several steps in a workflow or are working on a smaller browser screen, you may have more workflow steps than will fit in the configuration window. To navigate around the workspace and personalize it:

- Click and drag the workspace background to move the steps around the workspace. In this way you can reach steps at the top or bottom of the workflow that do not initially appear.
- Click the open button () to open the Workflow Definition dialog and the close button () to close the Workflow Definition dialog.
- Click the plus button with a circle around it () to add a new workflow step at that point in the workflow.
- Click the plus button in the lower left of the workspace () to zoom in on the steps.
- Click the minus button in the lower left of the workspace () to zoom out on the steps.
- Click the auto size button in the lower left of the workspace () to recenter and fit the steps to the window.

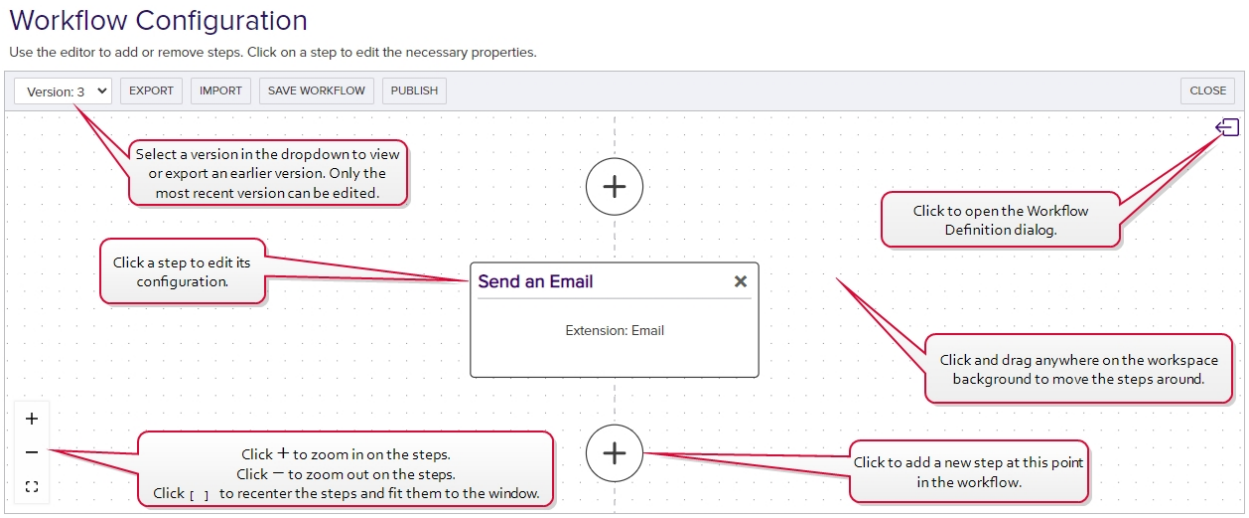


Figure 158: Using the Workflow Workspace

Tip: At any point while editing your workflow definition, you can click **Undo** at the bottom of the Add/Edit Workflow Definition dialog to undo changes made since the last save to the current workflow step you are editing or **Undo All** at the top of the workflow builder workspace to undo all changes made to the workflow definition since the last save.

Tip: To open a pop-out dialog with more real-estate for editing content in large text areas, like scripts and email messages:

- Navigate to the field you want to edit on the workflow definition.
- Click at the top right above the large text field.
- An *Edit Content* or *Edit PowerShell* window will open to accept your input. The *Edit Content* window supports token replacement. The *Edit PowerShell* window will open with a text editor. Enter your information.
- Click at the top right to close the edit window and return to the workflow definition, populated with your text.

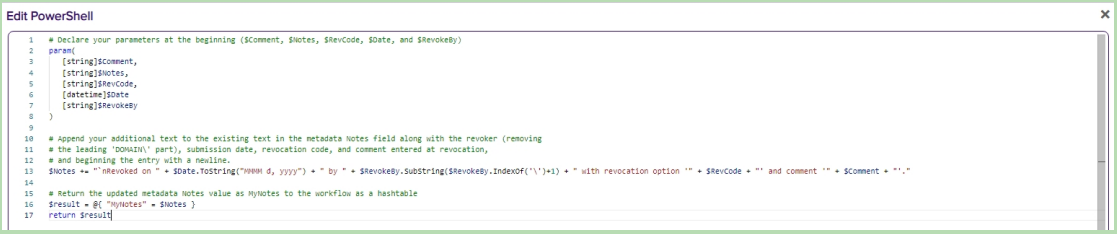


Figure 159: Edit PowerShell Window

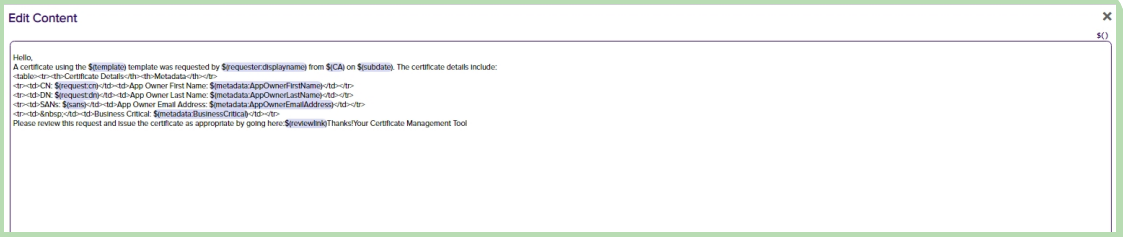


Figure 160: Edit Content Window

Adding, Copying or Modifying a Workflow Definition

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Definitions > Read
Workflows > Definitions > Modify

To add a new workflow definition or modify an existing one:

1. In the Management Portal, browse to *Workflow > Workflow Definitions*.
2. On the Workflow Definitions page, click **Add** from the top menu to create a new blank workflow definition, **Copy** from either the top or right click menu to copy an existing workflow to create a new one, or **Edit** from either the top or right click menu, to modify an existing one. This will open the workflow in the workflow builder workspace with the Workflow Definition dialog open on the right.

   Note: When you create a new workflow definition by copying an existing one, the word *copy* will be appended to the end of the definition name and the workflow key (template or certificate collection) will be cleared. Other data from the copied workflow will be retained.

3. In the Add/Edit Workflow Definition dialog on the Definition tab, enter a **Name** for your workflow.

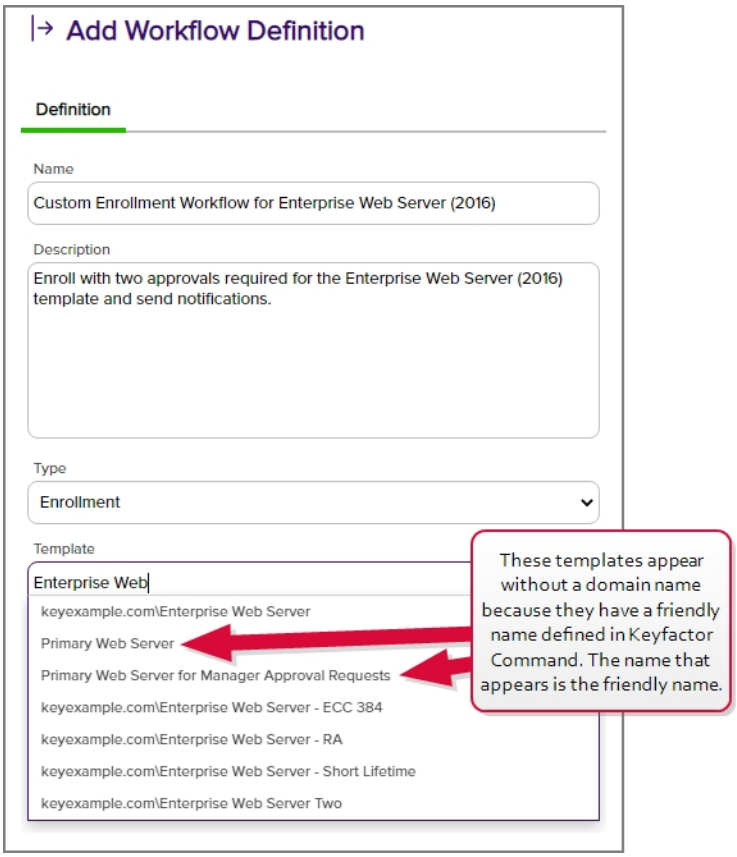


Figure 161: Create a New Workflow Definition

4. In the **Description** field, enter a description for the workflow definition.

5. In the **Type** dropdown, select the type of requests this workflow will handle. See Workflow Types on page 230 for a description of each workflow type.

   Note: This field cannot be modified on an edit.

6. Once you have selected a type, a **key field** will appear.

   - If you selected a type of Enrollment or Revocation, the key field is **Template**.
   - If you selected a type of Certificate Entered Collection or Certificate Left Collection, the key field is **Certificate Collection**.

   Begin typing in the **Template** or **Certificate Collection** field to search for available templates or certificate collections or click in the field and scroll down to locate your desired template or certificate collection. Templates that have been configured with a template friendly name will appear by friendly name.

Note: The key cannot be changed on an edit.

7. On the Workflow Configuration page, click the plus button in between two workflow steps where you want to add a new step. A new step box will be added below the plus that you clicked.

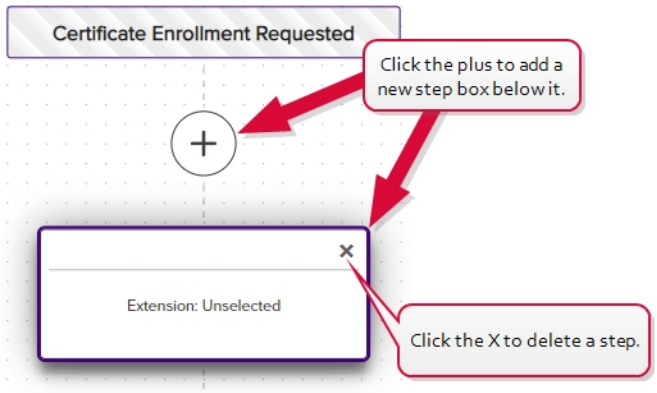


Figure 162: Click Plus to Add a New Workflow Definition Step

Tip: To delete a step, click the X at the top right of the step box and confirm that you want to delete the step.

8. Click the new step box to load the step in the Add/Edit Workflow Definition dialog. If the dialog is not already open, clicking a step will open it, or you can open a step by clicking the open button ( ) and then clicking the desired step to load it into the dialog.

9. In the Add/Edit Workflow Definition dialog on the Step tab in the General section, select a **Step Type** for the step in the dropdown. To narrow the list of step types in the dropdown, begin typing a search string in the Search field. See Workflow Steps on page 231 for a description of each step type.

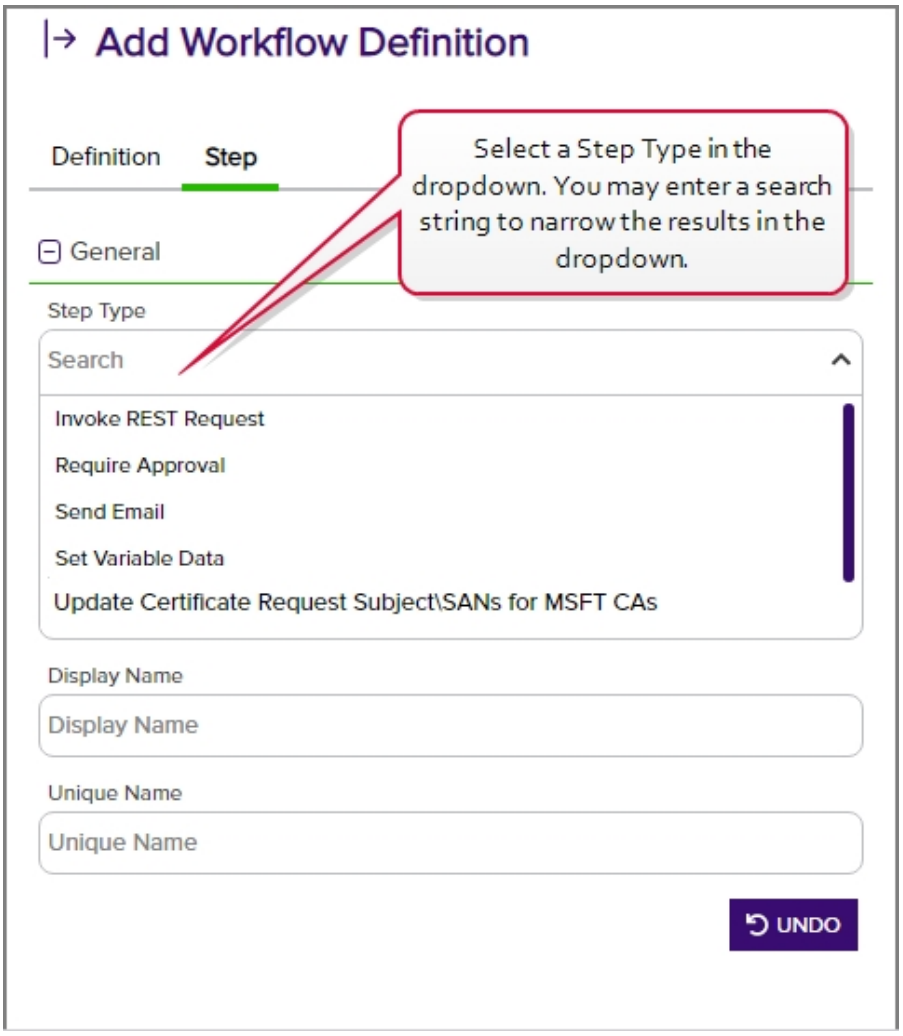


Figure 163: Select a Workflow Definition Step

Note: On an edit, if you change the workflow step type, you must also change the **Unique Name**. Changing the workflow step type without changing the unique name will result in an error similar to the following:

```
System.Collections.Generic.KeyNotFoundException: The given key was not present
in the dictionary
```

Instead of changing both the workflow step type and unique name, you may be prefer to delete the step and create a new step of the desired type.

10. In the Add/Edit Workflow Definition dialog on the Step tab in the General section, enter a **Display Name** for the step. This name appears as the title of the step box on the workflow workspace page.

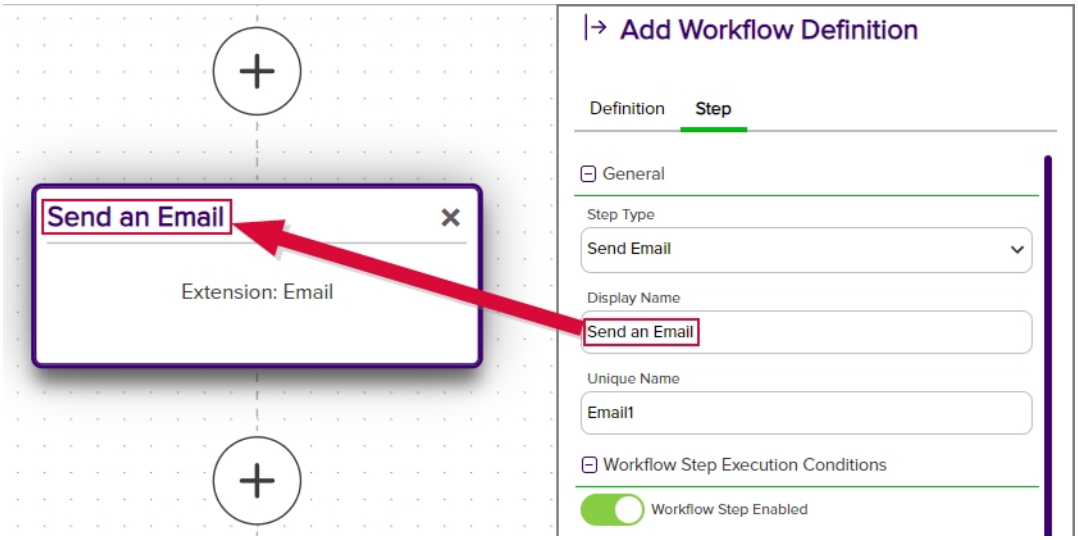


Figure 164: Display Name is Step Name Title

11. In the Add/Edit Workflow Definition dialog on the Step tab in the General section, either accept the automatically generated **Unique Name** for the step or modify it. This name must be unique among the steps within the particular workflow. It is intended to be used as a user-friendly reference ID.

12. In the Add/Edit Workflow Definition dialog on the Step tab in the Workflow Step Execution Conditions section, click the **Workflow Step Enabled** toggle to enable or disable the workflow. It is enabled by default.

13. Workflow Step Execution Conditions

    In the Workflow Step Execution Conditions section, click **Add** in the Optional Workflow Step Conditions for Execution section to create a new condition for the step. Conditions are true/-false statements indicating whether the step should run and can be based on tokens. See Workflow Step Execution Conditions on page 249 for in-depth information and examples of workflow step conditions.

14. Configuration Parameters

    The fields in the Configuration Parameters section will vary depending on the type of step you're configuring. See Workflow Definitions Configuration Parameters on page 251 for in-depth information and examples of each configuration parameter option.

15. For Require Approval steps or custom steps requiring signals, in the Workflow Step Editor in the Signals section, select one or more security roles (see Security Roles and Claims on page 622) in the **Approval Status** dropdown. To narrow the list of security roles in the dropdown, begin typing a search string in the Search field. Click the erase icon (◇) to clear your selections.

    Users who hold the security role(s) selected here will be able to submit signals (e.g. approve requests) for this workflow.

    Tip: Signals represent data used at the point in the workflow step where the workflow needs to continue based on user input. Here, you're configuring which users are allowed to provide that input.

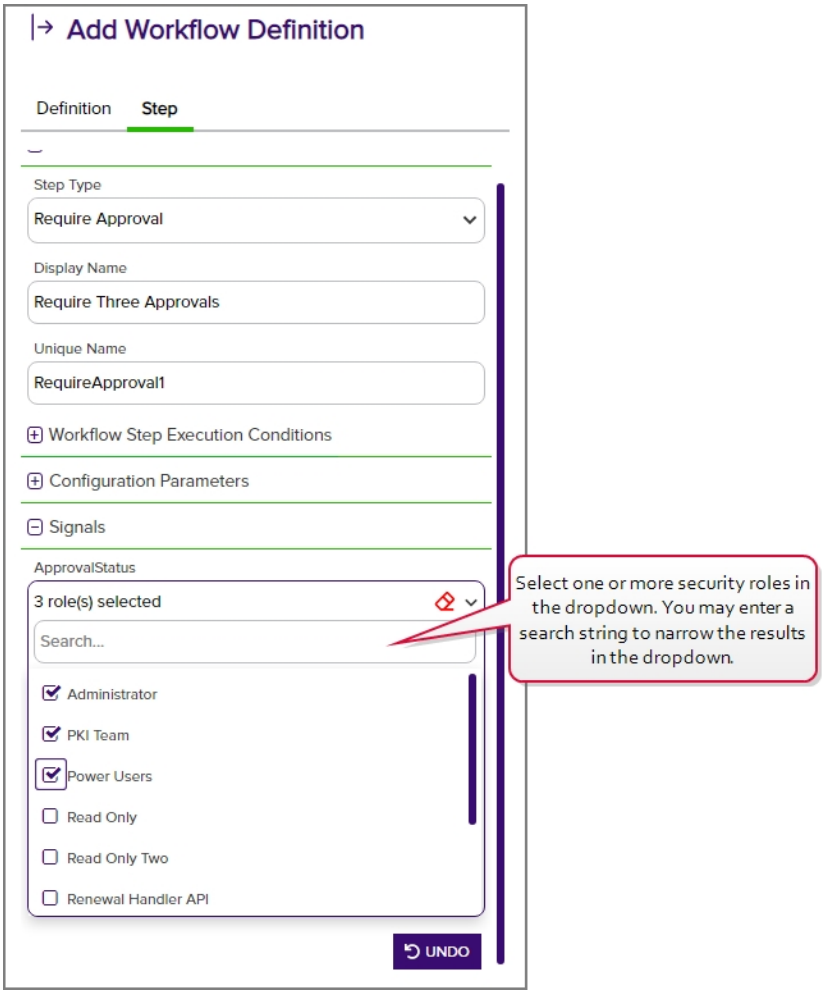


Figure 165: Signals Configuration for a Requires Approval Workflow Definition Step

Important: If all the security roles configured for a workflow step are deleted from Keyfactor Command, no users will be able to submit signals for workflow instances initiated with that workflow definition. To remedy this, update the workflow definition with one or more current security roles, re-publish it, and then restart any outstanding workflow instances.

16. Click **Save Workflow** at the top of the workflow workspace to save the workflow step.
17. On the Workflow Configuration page, click the plus button in between two workflow steps to add another step in the workflow or click **Save Workflow** to save the workflow with its current steps.
18. Before you can use the workflow, it must be published to activate it. Click the **Publish** button at the top of the workflow workspace to publish it immediately or return to the workflow definitions page and publish it later, if desired (see Publishing a Workflow Definition on the next page).

    Tip: Clicking **Publish** automatically saves the workflow.

19. To close the workflow workspace and return to the workflow definitions page, click the **Close** button at the top of the workflow workspace.

Note: If you edit an existing *published* workflow definition, a new version of the workflow definition will be created. If you edit an existing workflow definition which has *never been published*, the existing configuration will be overwritten with the changes you've made—a new version will not be created.

An audit log entry is created when you add or edit a workflow definition (see Audit Log on page 716).

Deleting a Workflow Definition

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Definitions > Read
Workflows > Definitions > Modify

To delete a workflow definition:

1. In the Management Portal, browse to *Workflow > Workflow Definitions*.
2. On the Workflow Definitions page, select a workflow definition and click **Delete** from either the top or right-click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Note: The built-in global workflow definitions (*Global Revocation Workflow* and *Global Enrollment Workflow*) cannot be deleted. A workflow definition cannot be deleted if there is an active or suspended workflow instance for the workflow definition.

An audit log entry is created when you delete a workflow definition (see Audit Log on page 716).

Publishing a Workflow Definition

Workflow definitions are drafts that cannot be actively used until you take the step to publish them. This allows you to add new workflows or update existing ones without interrupting the flow of current activity. Then, once the workflow definition is complete and ready for use, you can activate it. This can be done on the workflow workspace page while editing the workflow (see Adding, Copying or Modifying a Workflow Definition on page 237) or from the workflow definitions page.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Definitions > Read
Workflows > Definitions > Modify

To publish a workflow definition from the workflow definitions page:

1. In the Management Portal, browse to *Workflow > Workflow Definitions*.
2. On the Workflow Definitions page, select a workflow definition and click **Publish** from either the top or right-click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Alternately, publish a workflow definition from the workflow builder workspace:

1. In the Management Portal, browse to *Workflow > Workflow Definitions*.
2. On the Workflow Definitions page, select a workflow definition to open the workflow definition you wish to publish.
3. Click the **Publish** button at the top of the workflow workspace to publish it.

Exporting a Workflow

Workflow definitions can be exported either from the workflow workspace page while viewing or editing the workflow (see Adding, Copying or Modifying a Workflow Definition on page 237) or from the workflow definitions page.

- Export a workflow for backup purposes.
- Export a workflow that you've fully configured and which you need to replicate and then import under another name to create a duplicate of it.
- Export a previous version of a workflow and import it as the current version to revert to using the previous version.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Definitions > Read

To export a workflow definition from the workflow workspace:

1. In the Management Portal, browse to *Workflow > Workflow Definitions*.
2. On the Workflow Definitions page, click **Edit** from either the top or right click menu. This will open the workflow in the workflow workspace with the Workflow Definition dialog open on the right.
3. At the top of the workflow workspace, select a different **Version** of the workflow in the drop-down, if desired (see Workflow Versions on page 248).
4. At the top of the workflow workspace, click **Export**.
5. Browse to place the exported file on the local computer. The file will have an extension of .json.

To export a workflow definition from the workflow definitions page:

1. In the Management Portal, browse to *Workflow > Workflow Definitions*.
2. On the Workflow Definitions page, select a workflow definition and click **Export** from either the top or right-click menu.
3. In the Export Workflow Definition dialog, select a **Version** and click **Export**.

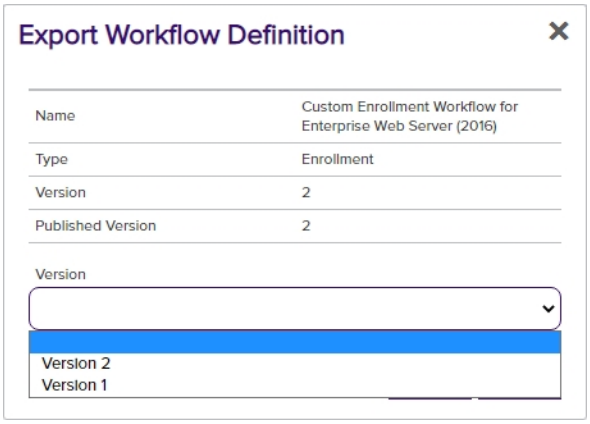


Figure 166: Export Workflow Definition

4. Browse to place the exported file on the local computer. The file will have an extension of .json.

Note: The following information is removed on export and will not be in the exported file:

- **Secrets**
  Some types of workflow steps include secret values (e.g. passwords). Secret values are not exported. If your workflow includes steps with secret values, these will need to be re-entered if you choose to import the exported file.
- **Roles for Signals**
  Some types of workflow steps make use of signals to allow users to provide input to the workflow midstream (e.g. provide approvals). This requires configuration of security roles that define who is allowed to provide this input. These security role values are not exported. You will need to set appropriate security roles on any workflow steps that use signals if you choose to import the exported file.

Importing a Workflow

Workflow definitions can be imported either to create a new workflow or to replace an existing workflow (e.g. to revert to a backup). When you import a workflow definition while editing an existing workflow definition, it will overwrite any changes you have made to the existing workflow since the last time it was published. Previously published versions of the workflow—including the most recent—will be retained. This is useful in cases where you want to export a previous version of a workflow and reimport it to make it the currently active version. This can be used to import a new workflow customized for you by the Keyfactor team.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Definitions > Read
Workflows > Definitions > Modify

To import a workflow definition:

1. In the Management Portal, browse to *Workflow > Workflow Definitions*.
2. On the Workflow Definitions page, click **Add** from the top menu to create a new workflow definition into which you will import, or **Edit** from either the top or right click menu, to import into an existing one to revert to a previous version. This will open the workflow in the workflow workspace with the Workflow Definition dialog open on the right.
3. At the top of the workflow workspace, click **Import**.
4. Browse to locate the workflow definition file you wish to import. Only files with an extension of *.json* will appear.

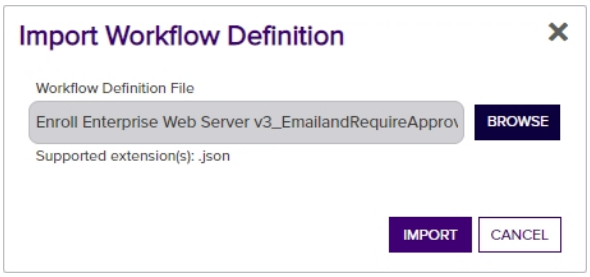


Figure 167: Browse to Locate a Workflow Definition to Import

Tip: In order to be successfully imported, the file must be correctly formatted JSON with at least *WorkflowType* and *Steps* properties. The maximum file upload size is 2 MB.

5. Click **Import** to import the workflow definition and populate it into the workflow workspace.
6. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.
7. In the workflow workspace, edit and save the workflow definition as needed as per Adding, Copying or Modifying a Workflow Definition on page 237. The following values will need attention:

    - **Key (Template or Certificate Collection)**

      When the workflow definition is imported into a new workflow definition, the key is cleared. You will need to set an appropriate key (template for enrollment or revocation type workflows, certificate collection for workflows of type certificate entered or left collection) on the imported workflow definition before saving. The key is not cleared for imports into workflows with existing published versions.

      This is done both to support export of workflow definitions from one environment and import into another where the key set likely would be different and to support copying of workflow definitions, since you can't have two definitions for the same key.
    - **Secrets**

      Some types of workflow steps include secret values (e.g. passwords). Secret values are not imported. If your workflow includes steps with secret values, these will need to be re-entered. This is true for imports into new workflow definitions and workflow definitions with existing published versions.
    - **Roles for Signals**

      Some types of workflow steps make use of signals to allow users to provide input to the workflow midstream (e.g. provide approvals). This requires configuration of security roles that define who is allowed to provide this input. These security role values are not imported. You will need to set appropriate security roles on any workflow steps that use signals before saving. This is true for imports into new workflow definitions and workflow definitions with existing published versions.

This is done to support export of workflow definitions from one environment and import into another where the security role set likely would be different.

Important: If you're importing a copy of a workflow definition that already exists in Keyfactor Command and you want to save it as a separate copy, be sure to change the **Name** of the workflow before saving the imported workflow to avoid overwriting the existing version of the workflow.

Workflow Versions

When you open a workflow definition for editing, you will see the version of the workflow shown at the upper left of the workflow workspace in a dropdown. By default, the current version will be shown.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Definitions > Read

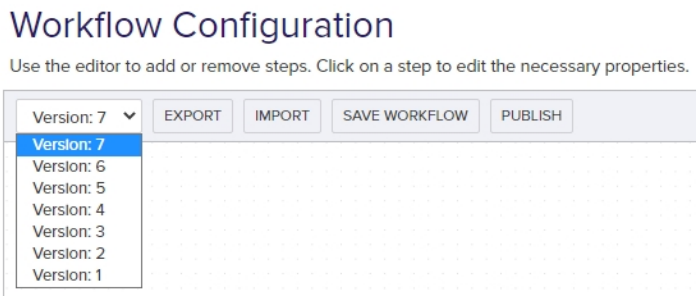


Figure 168: Workflow Definition Versions: View Current Version

When you have the current, most recent, version of the workflow loaded, you will see several options in the button bar at the top of the workflow workspace (if you have appropriate permissions) and the Add/Edit Workflow Definition Dialog will be active. If you select an older version in the dropdown, only the Version, Export, and Close options will appear on the workflow workspace button bar and the Add/Edit Workflow Definition Dialog will be read only.

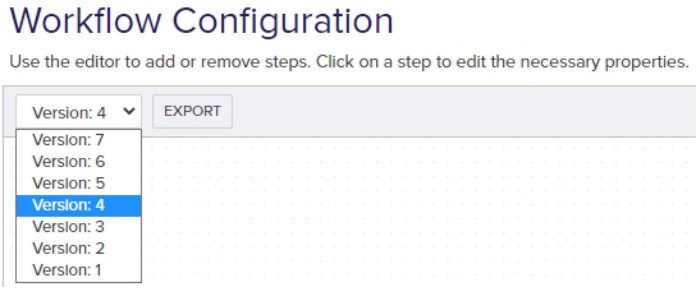


Figure 169: Workflow Definition Versions: View Previous Version

This option is designed to allow you to review previous versions of a workflow or export them as backups or to be re-imported to be used as a base for generating new workflows.

Workflow Step Execution Conditions

This section provides in-depth explanations and examples for using conditions in workflow definitions (see Workflow Step Execution Conditions on page 241).

Tip: Tokens (a.k.a. substitutable special text) may be used in the condition field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can create a token in a PowerShell step that has a value of True or False based on something determined in the step and then evaluate that token in a subsequent require approval step to determine whether to execute the require approval step based on the results from the PowerShell step. Fields that support tokens are indicated with $() at the top right of the field. To use a token in a field, begin typing at the location where you want the token to appear, starting with **$(**. Once you have typed **$(**, a second **)** will appear automatically along with a dropdown of available tokens to choose from. You may continue typing to narrow the values in the dropdown (e.g. type **$(req** to see only tokens that begin "req").

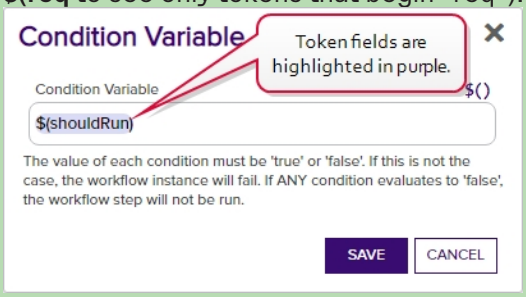


Figure 170: Tokens are Highlighted

To add a new condition, click Add and in the Condition Variable field enter either a static value of True or False or a token that will have a value of True or False at the time the step is run. More than one condition may be added. If multiple conditions are used in the same step, all conditions must have a value of True at the time the step is evaluated to be run in order for the step to run. If any single condition evaluates to False, the step will not run.

Set Variable Data and Require Approval with a Condition

The following example takes the common name entered during an enrollment and evaluates it to determine whether the domain name on it matches "keyexample.com" or not. If the domain is "keyexample.com", the enrollment is allowed to proceed without requiring approval. If the domain does not match "keyexample.com", the request requires approval. This example uses both a PowerShell Set Variable Data step and a Require Approval step.

To do this, first create the PowerShell step. Here we use a *Set Variable Data* step (see Set Variable Data on page 275) since no functions need to be called outside the confines of Keyfactor Command,

though you could use a *Custom PowerShell Script* step instead. Add a Script Parameter to pull the request CN into the script.

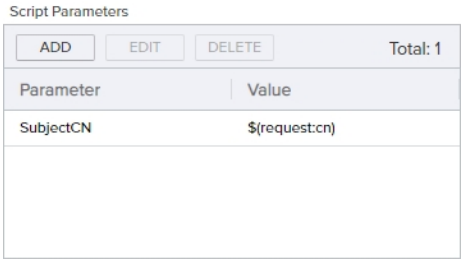


Figure 171: Conditions Example: Add Parameters

In the Insert PowerShell Script field, enter a script similar to the following:

```
# Declare your parameter at the beginning
param(
   [string]$SubjectCN
)

# Initialize a variable for the response
$shouldRun = @()

# Check to see if the requested CN ends with keyexample.com and require approval in the next step if
it does not
$Suffix = "keyexample.com"

if ($SubjectCN.EndsWith($Suffix))
{
   $shouldRun = "False"
}else {
   $shouldRun = "True"
}

# Return the true/false value to the workflow as a hashtable
$result = @{ "shouldRun" = $shouldRun; }
return $result
```

Next, create the require approval request step (see Require Approval on page 264) with $(shouldRun) as a condition like so:

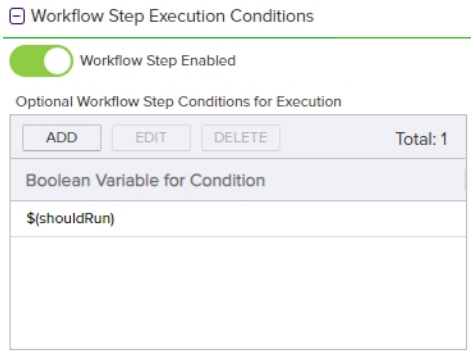


Figure 172: Conditions Example: Add Conditions for Require Approval Step

This condition on the require approval step will cause the approvals configured in the step to be required only if the CN submitted in the request does not end with "keyexample.com", so a request for "CN=mycert.keyother.com" will require approval but a request for "CN=mycert.keyexample.com" will not.

Workflow Definitions Configuration Parameters

This section provides in-depth explanations and examples for using configuration parameters in workflow definitions (see Configuration Parameters on page 241). The configuration parameters vary depending on the type of workflow step. The below information is broken down by step type. Most steps have at least one example, though the examples may incorporate more than one step type. The following examples are provided:

- Set Variable Data and Require Approval with a Condition on page 249
- Invoke REST Request for Data Lookup on Enrollment on page 255
- Invoke REST Request and Set Variable Data with Variable Passing on page 261
- Set Variable Data for Revocation on page 277
- Set Variable Data for Enrollment on page 279
- Use Custom PowerShell for Enrollment on page 283
- Use Custom PowerShell and Require Approval on page 286
- Use Custom PowerShell with Embedded REST Request, Send Email, and Require Approval on page 266
- Update Certificate Request Subject\SANS for MSFT CAs on page 291

For an overview of the types of workflow steps, see Workflow Steps on page 231.

Invoke REST Request

This step type is used only for Keyfactor Command implementations using Active Directory as an identity provider. If you're using an identity provider other than Active Directory, refer to the Invoke REST Request with OAuth step type (see Invoke REST Request with OAuth on page 258).

Tip: Tokens (a.k.a. substitutable special text) may be used in the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id). Fields that support tokens are indicated with $() at the top right of the field. To use a token in a field, begin typing at the location where you want the token to appear, starting with **$(**. Once you have typed **$(**, a second **)** will appear automatically along with a dropdown of available tokens to choose from. You may continue typing to narrow the values in the dropdown (e.g. type **$(req** to see only tokens that begin "req").

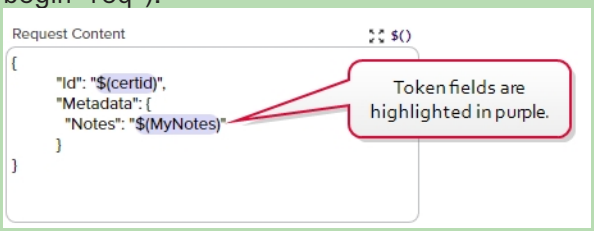


Figure 173: Tokens are Highlighted

- **Headers**: Enter any headers needed for your request. For a Keyfactor API request, this might look like:

```
x-keyfactor-requested-with: APIClient
x-keyfactor-api-version: 1
```

Tip: For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.

- **Variable to Store Response in**: Provide a name for the parameter in which to store the response data from your request. You can then reference this parameter from subsequent steps in the workflow.

Tip: The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result (0) and only the ClientMachine name, in the email message you would enter the following:

```
$(MyResponse.[0].ClientMachine)
```

- **Verb**: In the dropdown, select the type of request you wish to make (e.g. GET, POST).

- **Use Basic Authentication**: Check this box to use Basic authentication for the request. If you do not check this box, Windows authentication in the context of the Keyfactor Command application pool user will be used (see Create Service Accounts for Keyfactor Command on page 2757).
- **Username** and **[Password]**: Enter the username and password to use for authentication if *Use Basic Authentication* is checked. In the Username and Password dialogs, the options are Load From Keyfactor Secrets or Load From PAM Provider.
  A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

  Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

  Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

  Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

  ### CyberArk

  Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

  - Safe—The name of the safe the credential resides in.
  - Object—The name of the username or password object in the safe.
  - Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

  ### Delinea

  Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

  - Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
  - Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

- **URL**: Enter the request URL for the request, including tokens if desired. For a Keyfactor API request, this might look like (with query parameters):

  ```
  https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-
  contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL
  ```

  Or, with tokens:

  ```
  https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)
  ```

  Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example:

  ```
  192.168.12.0/24,192.168.14.22/24
  ```

  When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.

- **Content-Type**: In the dropdown, select the content type for the request:
  - application/json
- **Request Content**: The request body of the REST request, if required, with tokens, if desired. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):

```
{
   "Id": "$(certid)",
   "Metadata":{
      "RevocationComment":"$(cmnt)"
   }
}
```

Note: This example assumes you have a metadata field called *RevocationComment* (see Certificate Metadata on page 710).

For an example using a Keyfactor API endpoint within a PowerShell script, see Use Custom PowerShell with Embedded REST Request, Send Email, and Require Approval on page 266. For an example using an Invoke REST Request and a PowerShell step passing data between them, see Invoke REST Request and Set Variable Data with Variable Passing on page 261.

Invoke REST Request for Data Lookup on Enrollment

The following example takes the requester for an enrollment request and uses that to look up some information about the requester's SSH user record. This example uses a PowerShell step, a REST Request step, and an email step to deliver the information about the SSH user.

To do this, first create the PowerShell step. Here we use a *Set Variable Data* step (see Set Variable Data on page 275) since no functions need to be called outside the confines of Keyfactor Command other than those that are supported within Set Variable Data, though you could use a *Custom PowerShell Script* step instead. Add Script Parameters to pull the requester into the script (see Figure 174: Requester Lookup Example: Add Parameters).

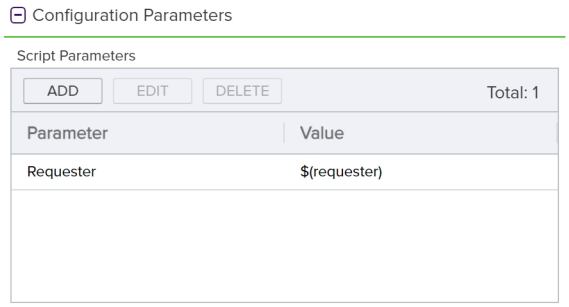


Figure 174: Requester Lookup Example: Add Parameters

In the Insert PowerShell Script field, enter a script similar to the following:

```
# Declare your parameters at the beginning
param(
   [string]$Requester
)

#Convert the requester to JSON
$RequesterJSON = ($Requester | ConvertTo-Json)

# Build the query for the REST step
$Query = "Username -eq $RequesterJSON"
```

```
# Return the query to the workflow as a hashtable
$result = @{ "Query" = $Query }
return $result
```

Next, create the REST request step with the following values:

- **Headers**: The GET /SSH/Users method has a version 2 with additional features. To access version 2, this version must be specified in the header.

  ```
  {
     "x-keyfactor-requested-with": [
        "APIClient"
     ],
     "x-keyfactor-api-version": [
        "2"
     ]
  }
  ```

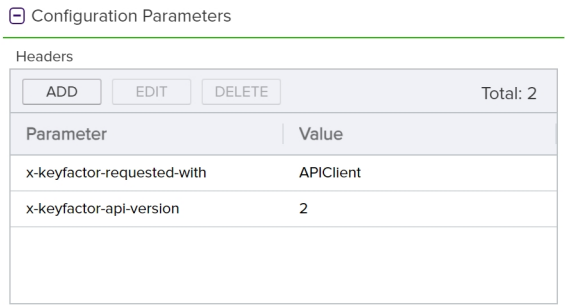
  

  Figure 175: Requester Lookup Example: Add Headers for REST Request

- **Variable to Store Response in**: MyResponse
- **Verb**: GET
- **URL**: Note that the $(Query) from the PowerShell step is provided in the URL (*keyfactor.keyexample.com* is your Keyfactor Command server name).

  ```
  https://keyfactor.keyexample.com/KeyfactorAPI/SSH/Users?QueryString=$(Query)
  ```

- **Content-Type**: application/json
- **Request Content**: None (The request is provided in the URL; there is no body for this endpoint.)

This REST step takes the Query built in the PowerShell step, containing the requester's username, and submits it to the GET /SSH/Users endpoint. The response contains all the information from the user's SSH key record.

Next, create the Send Email step. In the configuration parameters, give your email a Subject that will help highlight the information:

```
Certificate Enrollment Request for $(request:cn)
```

In the main Message of the email, provide the required information using tokens, including the information retrieved from the GET /SSH/Users endpoint (see Substitutable Text Tokens for Workflow on page 295):

```
Hello,

A certificate using the $(template) template was requested by $(requester:displayname) from $(CA) on
$(subdate).

The certificate details include:

   <ul>
      <li>CN: $(request:cn)</li>
      <li>DN: $(request:dn)</li>
      <li>SANs: $(sans)</li>
      <li>App Owner First Name: $(metadata:AppOwnerFirstName)</li>
      <li>App Owner Last Name: $(metadata:AppOwnerLastName)</li>
   </ul>

The requester's SSH information includes:

   <ul>
      <li>ID: $(MyResponse.Result[0].Key.Id)</li>
      <li>Email: $(MyResponse.Result[0].Key.Email)</li>
      <li>Comments: $(MyResponse.Result[0].Key.Comments.[*])</li>
   </ul>

Thanks!
Your Certificate Management Tool
```

Note: The $(requester:displayname) substitutable special text token is only supported in environments using Active Directory as an identity provider.

In the Recipients, add all the email recipients who should receive this information.

Note: If your REST request takes a long time to complete, the step may time out and the workflow instance fail. The default timeout is 60 seconds and is configurable with the *Workflow Step Run Timeout* application setting (see Application Settings: Workflow Tab on page 621).

Invoke REST Request with OAuth

This step type is used only for Keyfactor Command implementations using an identity provider other than Active Directory. If you're using Active Directory, refer to the Invoke REST Request step type (see Invoke REST Request on page 251).

Tip: Tokens (a.k.a. substitutable special text) may be used in the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id). Fields that support tokens are indicated with $() at the top right of the field. To use a token in a field, begin typing at the location where you want the token to appear, starting with **$(**. Once you have typed **$(**, a second **)** will appear automatically along with a dropdown of available tokens to choose from. You may continue typing to narrow the values in the dropdown (e.g. type **$(req** to see only tokens that begin "req").

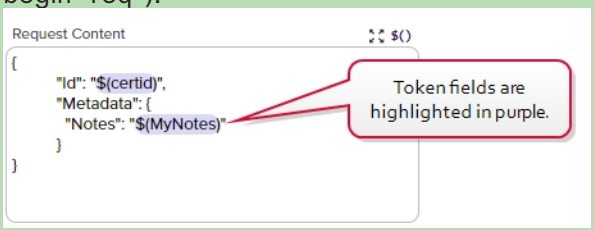


Figure 176: Tokens are Highlighted

- **Headers**: Enter any headers needed for your request. For a Keyfactor API request, this might look like:

```
x-keyfactor-requested-with: APIClient
x-keyfactor-api-version: 1
```

Tip: For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.

- **Variable to Store Response in**: Provide a name for the parameter in which to store the response data from your request. You can then reference this parameter from subsequent steps in the workflow.

> **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result (0) and only the ClientMachine name, in the email message you would enter the following:
>
> ```
> $(MyResponse.[0].ClientMachine)
> ```

- **Verb**: In the dropdown, select the type of request you wish to make (e.g. GET, POST).
- **Client ID**: Enter the ID of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844).
- **Client Secret**: Click the **Set\UpdateClient Secret** button and in the Client Secret dialog, enter the secret for the identity provider client that should be used to authenticate the session. In the Client Secret dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

  > **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

  Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

  Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

CyberArk

Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

- Safe—The name of the safe the credential resides in.
- Object—The name of the username or password object in the safe.

- Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

- **Token Endpoint**: The URL of the token endpoint for your identity provider instance. For example:

  ```
  https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token
  ```

  For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

- **URL**: Enter the request URL for the request, including tokens if desired. For a Keyfactor API request, this might look like (with query parameters):

  ```
  https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL
  ```

  Or, with tokens:

  ```
  https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)
  ```

  Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example:

```
192.168.12.0/24,192.168.14.22/24
```

When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.

- **Content-Type**: In the dropdown, select the content type for the request:
  - application/json
- **Request Content**: The request body of the REST request, if required, with tokens, if desired. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):

```
{
   "Id": "$(certid)",
   "Metadata":{
      "RevocationComment":"$(cmnt)"
   }
}
```

Note: This example assumes you have a metadata field called *RevocationComment* (see Certificate Metadata on page 710).

For an example using a Keyfactor API endpoint within a PowerShell script, see Use Custom PowerShell with Embedded REST Request, Send Email, and Require Approval on page 266. For an example using an Invoke REST Request to do a data lookup on enrollment, see Invoke REST Request for Data Lookup on Enrollment on page 255.

Invoke REST Request and Set Variable Data with Variable Passing

The following example takes the revocation comment entered when a certificate is revoked and puts it together with some other information into a custom metadata field, retaining any existing data in that metadata field. This example uses both a PowerShell step and a REST Request step to demonstrate passing of information from one step to the other.

To do this, first create the PowerShell step. Here we use a *Set Variable Data* step (see Set Variable Data on page 275) since no functions need to be called outside the confines of Keyfactor Command, though you could use a *Custom PowerShell Script* step instead. Add Script Parameters to pull the revocation comment, submission date, revocation code, user making the revocation request, and the metadata field into which you will place your updated comment (*Notes* in this example) into the script. Figure 177: Metadata Update Example: Add Parameters shows only four of these. The metadata field *Notes* is a BigText type field in this example (see Metadata Field Operations on page 710).

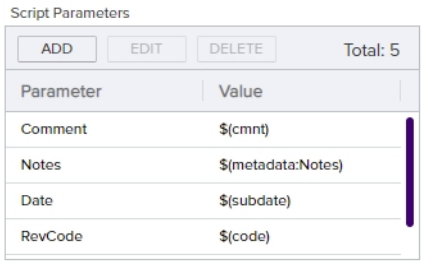


Figure 177: Metadata Update Example: Add Parameters

In the Insert PowerShell Script field, enter a script similar to the following:

```
# Declare your parameters at the beginning ($Comment, $Notes, $RevCode, $Date, and $RevokeBy)
param(
   [string]$Comment,
   [string]$Notes,
   [string]$RevCode,
   [datetime]$Date
   [string]$RevokeBy
)

# Append your additional text to the existing text in the metadata Notes field along with the revoker
(removing
# the leading 'DOMAIN\' part), submission date, revocation code, and comment entered at revocation,
# and beginning the entry with a newline.
$Notes += "`nRevoked on " + $Date.ToString("MMMM d, yyyy") + " by " + $RevokeBy.SubString
($RevokeBy.IndexOf('\')+1) + " with revocation option '" + $RevCode + "' and comment '" + $Comment +
"'."

# Return the updated metadata Notes value as MyNotes to the workflow as a hashtable
$result = @{ "MyNotes" = $Notes }
return $result
```

Next, create the REST request step with the following values:

- **Headers**: The API version does not need to be stated since version 1 is the default.

  ```
  {
     "x-keyfactor-requested-with": [
        "APIClient"
     ]
  }
  ```

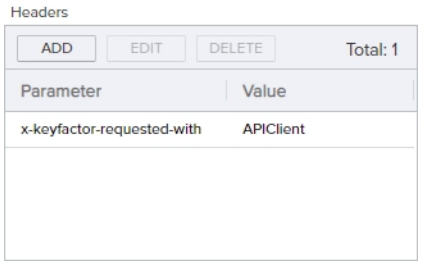


Figure 178: Metadata Update Example: Add Headers for REST Request

- **Variable to Store Response in**: None (there is no output from this command on a success)
- **Verb**: PUT
- **Client ID**: The client ID from your identity provider implementation (see Authenticating to the Keyfactor API on page 844). For example:

  ```
  Keyfactor-API-Workflow-User
  ```

- **Client Secret**: The client secret from your identity provider implementation (see Authenticating to the Keyfactor API on page 844). For example:

  ```
  WDBvGypDWuquyOmQQneeQp4IvmPDebz4
  ```

- **Token Endpoint**: The token endpoint from your identity provider implementation. For example:

  ```
  https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token
  ```

- **URL**: (Where *keyfactor.keyexample.com* is your Keyfactor Command server name.)

  ```
  https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/Metadata
  ```

- **Content-Type**: application/json
- **Request Content**:

  ```
  {
     "Id": "$(certid)",
     "Metadata": {
        "Notes": "$(MyNotes)"
     }
  }
  ```

This REST step takes the MyNotes output from the PowerShell step and updates the metadata Notes field to match that value. The resulting value in your Notes field will look something like this (assuming lines one, two and three were preexisting):

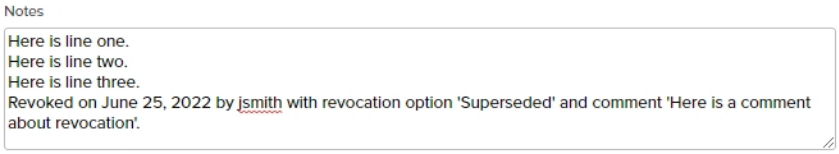


Figure 179: Metadata Update Example: Results

Note: You can achieve this same result of updating a metadata field entirely within PowerShell without using the REST step. This example uses both PowerShell and REST steps to demonstrate passing a value from one to the other.

Note: If your REST request takes a long time to complete, the step may time out and the workflow instance fail. The default timeout is 60 seconds and is configurable with the *Workflow Step Run Timeout* application setting (see Application Settings: Workflow Tab on page 621).

Require Approval

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester). Fields that support tokens are indicated with $() at the top right of the field. To use a token in a field, begin typing at the location where you want the token to appear, starting with **$(**. Once you have typed **$(**, a second **)** will appear automatically along with a dropdown of available tokens to choose from. You may continue typing to narrow the values in the dropdown (e.g. type **$(req** to see only tokens that begin "req").

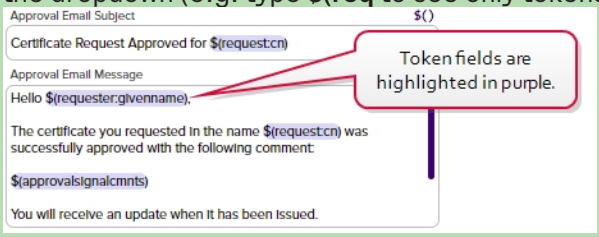


Figure 180: Tokens are Highlighted

Note: The users who will approve or deny the request must be members of a security role that is allowed to submit signals (e.g. approve requests) for the workflow in order to approve or deny the request.

- **Minimum Approvals**: Enter the minimum number of users who must approve the request to consider the request approved.

- **Denial Email Subject**: Enter the subject line for the email message that will be delivered if the request is denied, including tokens if desired.
- **Denial Email Message**: Enter the email message that will be delivered if the request is denied. The email message can be made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens.
- **Denial Email Recipients**: Click **Add**, enter a recipient for the denial email, and **Save**. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include:
    - $(requester:mail)

      The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.

      Note: The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider.

    - Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).
- **Approval Email Subject**: Enter the subject line for the email message that will be delivered if the request is approved, including tokens if desired.
- **Approval Email Message**: Enter the email message that will be delivered if the request is approved. The email message can be made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens.
- **Approval Email Recipients**: Click **Add**, enter a recipient for the approval email, and **Save**. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include:
    - $(requester:mail)

      The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.

      Note: The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider.

    - Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).

Tip: The approval message is delivered before the enrollment actually takes place. To send an email alerting interested parties that the certificate was issued, including a link to download the certificate, use an issued certificate alert (see Issued Certificate Request Alerts on page 188).

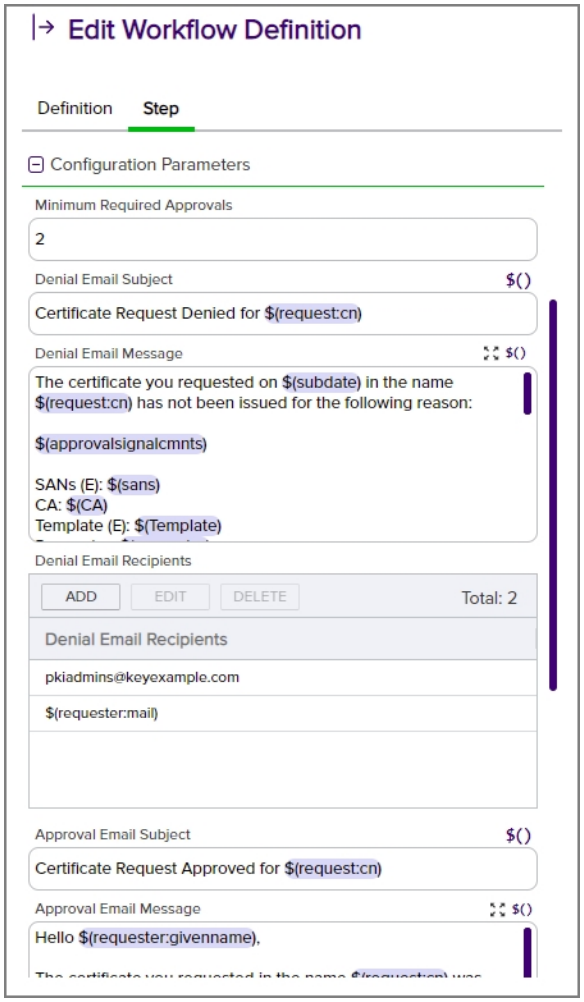


Figure 181: Configuration Parameters for a Require Approval Workflow Definition Step

For an example using a require approval step with a condition, see Set Variable Data and Require Approval with a Condition on page 249.

Use Custom PowerShell with Embedded REST Request, Send Email, and Require Approval

The following example takes information from an enrollment request that is destined for a certificate store and which requires approval and delivers it in an email to the managers designated as approvers of the request to provide them with the information necessary to make the go/no go

decision. This example uses a Custom PowerShell step that calls a Keyfactor API method, a Send Email step, and a Require Approval step.

This step needs to be a Use Custom PowerShell step rather than a Set Variable Data step because it calls a Keyfactor API method in a loop within the script. Since the API method needs to be called in a loop to retrieve multiple pieces of data, it's not practical to do this as a separate Invoke REST Request step.

To do this, first outside of Keyfactor Command create your PowerShell script containing content similar to the following:

```
# Declare your parameters at the beginning
param(
   [string]$StoreData,
   [string]$TimeData
)

# Pick one authentication mechanism for the Keyfactor API
# Basic authentication credentials to authenticate to the Keyfactor API
$user = 'KEYEXAMPLE\APIUser'
$pass = 'APIUserSuperSecretPassword'

# Encode Basic authentication credentials
$pair = "$($cred.Username):$($cred.GetNetworkCredential().Password)"
$encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
$basicAuthValue = "Basic $encodedCreds"

# Token authentication credentials to authenticate to the Keyfactor API (uncomment the below 9 lines
and update appropriately to use token authentication)
#$TokenBody = @{
#   grant_type = "client_credentials"
#   client_id = "Keyfactor-API-Workflow-User"
#   client_secret = "WorkflowAPIClientSecret"
#}
#$TokenHeaders = @{
#   'Content-Type' = 'application/x-www-form-urlencoded'
#}
#$TokenURL = "https://appsrvr18.keyexample.com:1443/realms/Keyfactor/protocol/openid-connect/token"

# Request token from Keyfactor Identity Provider for token authentication (uncomment the below line
to use token authentication)
#$TokenValue = Invoke-RestMethod -Method Post -Uri $TokenURL -Headers $TokenHeaders -Body $TokenBody

# Pick an authentication type for the header
$headerAuth = $basicAuthValue
```

```
#$headerAuth = "Bearer " + $TokenValue.access_token

# Build the headers for the API request
$headers = @{
   "Authorization"=$headerAuth
   "Accept"="application/json"
   "x-keyfactor-requested-with"="APIClient"
}

# Convert incoming data from JSON
$StoreDataInfo = $StoreData | ConvertFrom-Json
$TimeDataInfo = $TimeData | ConvertFrom-Json

# Initialize Variables introduced in script
$JobTime = ""
$JobTimeHuman = ""
$TimeMessage = ""
$StoreMessage = ""
$StoreList = New-Object System.Collections.Generic.List[System.Object]
$CommandServer = 'keyfactor.keyexample.com'

# Evalulate the scheduled delivery time for the certificate store management job, if present, and
populate a return message based on the content
if ("Immediate" -in $TimeDataInfo.PSobject.Properties.Name)
{
   $TimeMessage = "The certificate has been scheduled to be delivered at the next orchestrator run."
}
elseif ("ExactlyOnce" -in $TimeDataInfo.PSobject.Properties.Name)
{
   # Convert the certificate store job time to a human readable date/time
   $JobTime = [DateTime]($TimeDataInfo.ExactlyOnce.Time)
   $JobTimeHuman = $JobTime.ToString("M/dd/yyyy h:mm tt")
   $TimeMessage = "The certificate has been scheduled to be delivered at the following time:
$JobTimeHuman."
}
else
{
   $TimeMessage = " "
}

# Pull out the display names of the certificate stores based on the store IDs in the incoming store
info
foreach ($Store in $StoreDataInfo){
```

```
   $response = Invoke-WebRequest -Uri "https://$($Com-
mandServer)/KeyfactorAPI/CertificateStores/$($store.StoreId)" -Method:Get -Headers $headers -
ContentType "application/json" -ErrorAction:Stop -TimeoutSec 60
   $responseContent = $response.Content | ConvertFrom-Json

   #Add the store to the store list
   $StoreList.Add($responseContent.DisplayName)
}

# Evaluate whether any stores were returned and build a message that includes the store list if so,
and an alternate message if not
if ($StoreList.Count -gt 0)
{
   $StoreMessage = "If approved, the certificate will be delivered to the following stores:<br /><br
/>" + ($StoreList -join "<br />")
}
else
{
   $StoreMessage = "If approved, the certificate will be available for download in the Keyfactor
Command Management Portal. It has not been scheduled for delivery to any certificate stores."
}

# Return the time message and store message to the workflow as a hashtable
$result = @{ "TimeMessage" = $TimeMessage; "StoreMessage" = $StoreMessage }
return $result
```

Note: The method of providing authentication to Keyfactor Command for the Keyfactor API request in this script will vary depending on the authentication configuration of your Keyfactor Command implementation. The above script includes both Basic authentication (for environments using Active Directory as an identity provider) and Token authentication (for environments using an identity provider other than Active Directory) for demonstration purposes, but you should use only one type of authentication. For more information on authenticating to the Keyfactor API, see Authenticating to the Keyfactor API on page 844).

Next, use the *POST /Extension/Scripts* API endpoint (see POST Extensions Scripts on page 1709) to import your PowerShell script into the Keyfactor Command database before beginning to edit your workflow.

Once your PowerShell script has been imported into the Keyfactor Command database, you may begin creating your workflow. To create the Use Custom PowerShell step, add *Script Parameters* to pull the certificate store data—*$(Stores)*—and schedule time for the management job to add the certificate to the certificate stores—*$(ManagementJobTime)*—into the script as shown in Figure 182: Use Custom PowerShell with Embedded REST Request: Add Parameters. The *$(Stores)* and *$(ManagementJobTime)* tokens are not among those that appear in the dropdown.

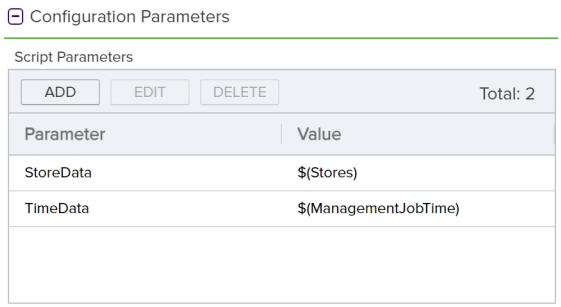


Figure 182: Use Custom PowerShell with Embedded REST Request: Add Parameters

In the PowerShell Field Name field, in the dropdown select the script you uploaded to the database.

Next, create the Send Email step. In the configuration parameters, give your email a Subject that will help highlight the request:

```
ACTION REQUIRED: Certificate Enrollment Request for $(request:cn)
```

This tells your users that they need to do something and includes the CN of the requested certificate in the subject line.

In the main Message of the email, tell your users what they need to do and provide the information from the request that will allow them to make an informed decision using tokens (see Substitutable Text Tokens for Workflow on page 295) and the certificate store information returned from the PowerShell script:

```
Hello,

A certificate using the $(template) template was requested by $(requester:displayname) from $(CA) on
$(subdate). $(StoreMessage)

$(TimeMessage)

The certificate details include:

   <ul>
      <li>CN: $(request:cn)</li>
      <li>DN: $(request:dn)</li>
      <li>SANs: $(sans))</li>
      <li>App Owner First Name: $(metadata:AppOwnerFirstName)</li>
      <li>App Owner Last Name: $(metadata:AppOwnerLastName)</li>
   </ul>

Please review this request and issue the certificate as appropriate by going here:
```

```
$(reviewlink)

Thanks!
Your Certificate Management Tool
```

Note: The $(requester:displayname) substitutable special text token is only supported in environments using Active Directory as an identity provider.

In the Recipients, add all the email recipients who could possibly approve or deny the request.

Now create your Require Approval step. The step does not need any Conditions. In the Configuration Parameters, enter a value of at least 1 for the Minimum Required Approvals. Give the Denial Email Subject a value something like:

```
Certificate Enrollment Request Denied for $(request:cn)
```

And the Approval Email Subject a value something like:

```
Certificate Enrollment Request Approved for $(request:cn)
```

Enter an appropriate message for the Denial Email Message. You may use tokens, including the returned values from the PowerShell script. You may want to deliver this message to the requester, so a message similar to this might be appropriate:

```
Hello $(requester:givenname),

The certificate you requested on $(subdate) in the name $(request:cn) has not been issued for the
following reason:

$(approvalsignalcmnts)

The certificate details include:

   <ul>
      <li>CN: $(request:cn)</li>
      <li>DN: $(request:dn)</li>
      <li>SANs: $(sans))</li>
      <li>App Owner First Name: $(metadata:AppOwnerFirstName)</li>
      <li>App Owner Last Name: $(metadata:AppOwnerLastName)</li>
```

```
   </ul>

For assistance, please contact <a href="mailto:support@keyexample.com">support@keyexample.com</a>.

Thanks!
Your Certificate Management System
```

Note: The $(requester:givenname) substitutable special text token is only supported in environments using Active Directory as an identity provider.

Enter an appropriate message for the Approval Email Message. This message would also likely go to the requester and might look similar to:

```
Hello $(requester:givenname),

The certificate you requested in the name $(request:cn) on $(subdate) was successfully approved with
the following comment:

$(approvalsignalcmnts)

The certificate details include:

   <ul>
      <li>CN: $(request:cn)</li>
      <li>DN: $(request:dn)</li>
      <li>SANs: $(sans))</li>
      <li>App Owner First Name: $(metadata:AppOwnerFirstName)</li>
      <li>App Owner Last Name: $(metadata:AppOwnerLastName)</li>
   </ul>

You will receive an update when it has been issued. For assistance, please contact <a href-
f="mailto:support@keyexample.com">support@keyexample.com</a>.

Thanks!
Your Certificate Management System
```

Note: The $(requester:givenname) substitutable special text token is only supported in environments using Active Directory as an identity provider.

In the Recipients for both the approval and denial emails, enter the token for the requester's email—*$(requester:mail)*.

Note: The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider.

In the Signals for the Require Approval step, select the security role(s) to which the users responsible for approving or denying the request belong.

Finish off the workflow process by configuring an issued certificate request alert to let the requester know when the certificate has been issued (see Issued Certificate Request Alerts on page 188).

Send Email

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester). Fields that support tokens are indicated with $() at the top right of the field. To use a token in a field, begin typing at the location where you want the token to appear, starting with **$(**. Once you have typed **$(**, a second **)** will appear automatically along with a dropdown of available tokens to choose from. You may continue typing to narrow the values in the dropdown (e.g. type **$(req** to see only tokens that begin "req").

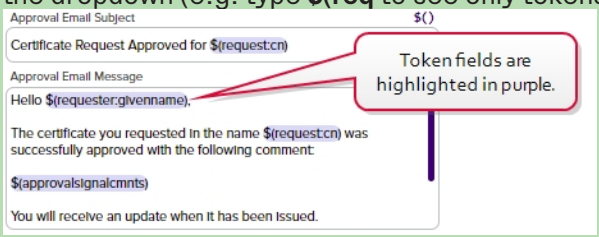


Figure 183: Tokens are Highlighted

- **Subject**: Enter the subject line for the email message that will be delivered when the workflow definition step is executed, including tokens if desired.
- **Message**: Enter the email message that will be delivered when the workflow definition step is executed. The email message can be made up of regular text and tokens. If desired, you can format the message body using HTML. For example, for an enrollment pending request notification:

  Hello,
  A certificate using the $(template) template was requested by $(requester:displayname) from $(CA) on $(subdate). The certificate details include:
  <table>
  <tr><th>Certificate Details</th><th>Metadata</th></tr>
  <tr><td>CN: $(request:cn)</td><td>App Owner First Name: $(metadata:AppOwnerFirstName)</td></tr>
  <tr><td>DN: $(request:dn)</td><td>App Owner Last Name:

$(metadata:AppOwnerLastName)</td></tr>
<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:AppOwnerEmailAddress)</td></tr>
<tr><td> </td><td>Business Critical: $(metadata:BusinessCritical)</td></tr>
Please review this request and issue the certificate as appropriate by going here:
$(reviewlink)
Thanks!
Your Certificate Management Tool

See Table 11: Tokens for Workflow Definitions for a list of available tokens.

Note: The $(requester:displayname) substitutable special text token is only supported in environments using Active Directory as an identity provider.

- **Recipients**: Click **Add**, enter a recipient for the email, and **Save**. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include:
  - $(requester:mail)

    The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.

    Note: The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider.

  - Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).

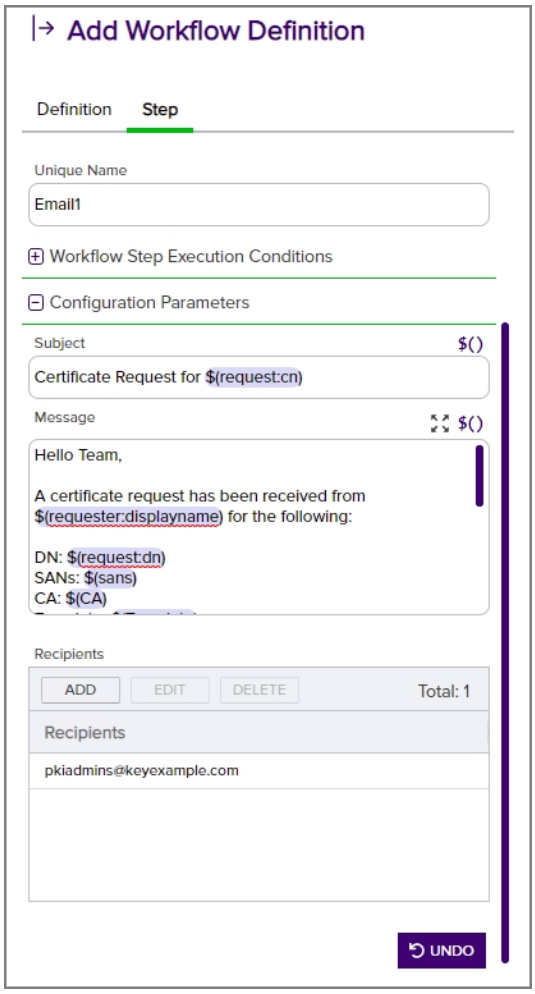


Figure 184: Step Configuration for an Email Workflow Definition Step

For an example using Send Email in a Require Approval workflow, see Use Custom PowerShell with Embedded REST Request, Send Email, and Require Approval on page 266.

Set Variable Data

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append additional data to it using PowerShell.

- **Script Parameters**: Add any parameters you will use to pass data into your script. These can contain static values or tokens (see Table 11: Tokens for Workflow Definitions). To add a parameter:

1. In the Script Parameters section, click **Add**.
2. In the Add/Edit Parameter dialog, enter a name for the parameter in the **Parameter** field. In the **Value** field, enter either a static value to be passed into the PowerShell script or select from the available tokens to pass the token value into the PowerShell in your parameter.
3. Click **Save** to save your parameter.

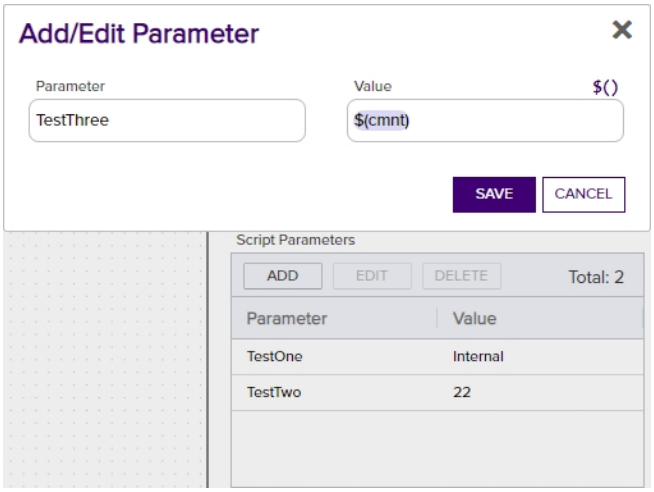


Figure 185: Add Parameters for PowerShell

- **Insert PowerShell Script**: Enter the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

  To receive your defined parameters from the previous step into the PowerShell script, begin the script by declaring the expected parameters like so (referencing the three parameters—TestOne, TestTwo, and TestThree):

  ```
  param(
     [string]$TestOne,
     [int]$TestTwo,
     [string]$TestThree
  )
  ```

  You may then use these parameters within the script.

  To return data from the PowerShell script, create a hashtable of the data you wish to return like so (where $MyField1 and $MyField2 are parameters introduced within the script and the new value in $TestThree is reloaded back into that parameter and used to update that field if the original parameter was set to a token):

```
$result = @{ "MyFieldOne" = $MyField1; "MyFieldTwo" = $MyField2; "TestThree" = $TestThree }
return $result
```

This will result in the following dictionary entries being added to the database and available for output or use in subsequent steps in the workflow:

```
{["MyFieldOne", "[your value as defined in the script"], ["MyFieldTwo", "[your value as defined
in the script"], ["TestThree", "[your value as defined in the script"],]}
```

You can reference these as tokens in subsequent steps as follows: $(MyFieldOne), $(MyFieldTwo), $(TestThree).

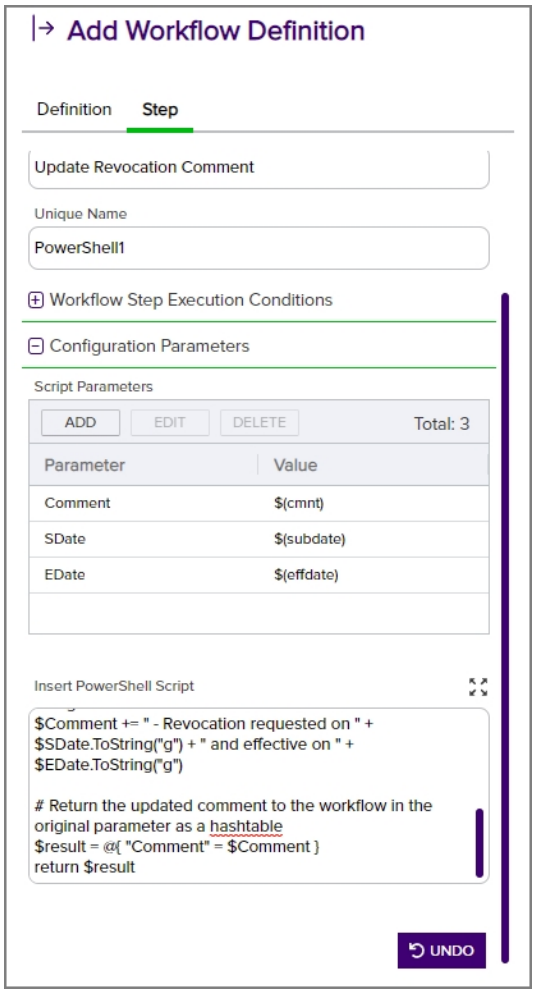


Figure 186: Configuration Parameters for a Set Variable Data Workflow Definition Step

Set Variable Data for Revocation

The following example takes the revocation comment entered when a certificate is revoked and appends an additional comment, including dates, to it. To create this, add Script Parameters to pull the revocation comment, submission date and effective date into the Set Variable Data script as shown in Figure 177: Metadata Update Example: Add Parameters.

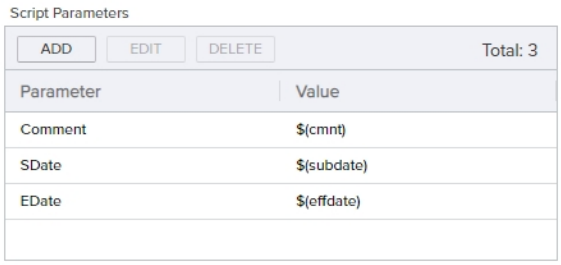


Figure 187: Revocation Comment Update Example: Add Parameters

In the Insert PowerShell Script field, enter a script similar to the following:

```
# Declare your parameters at the beginning ($Comment, $SDate, and $Edate)
param(
   [string]$Comment,
   [datetime]$SDate,
   [datetime]$EDate
)

# Append your additional text to the existing comment along with the submission and effective dates
$Comment += " - Revocation requested on " + $SDate.ToString("g") + " and effective on " +
$EDate.ToString("g")

# Return the updated comment to the workflow in the original parameter as a hashtable
$result = @{ "Comment" = $Comment }
return $result
```

The resulting comment will look something like:

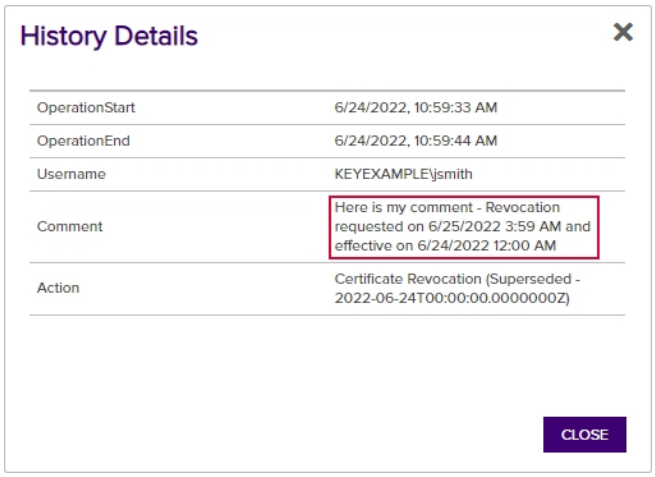


Figure 188: Revocation Comment Update Example: Results

You may reference the updated comment using the standard revocation comment token ($(cmnt)) in subsequent steps in your workflow and may view the updated comment wherever the revocation comment is available for viewing within Keyfactor Command.

Set Variable Data for Enrollment

The following example takes two additional enrollment fields submitted on an enrollment and sets the value of one to a fixed value if the value of the other (a multi-value field) is a given value using a Set Variable Data script. In other words, the possible values for Department (a multi-value field) are:

- Accounting
- E-Commerce
- HR
- IT
- Marketing
- R & D
- Sales

If the value of Department is anything other than Accounting, the value of Code (a string field) can be any value. If the value of Department is Accounting, anything submitted in the Code field by the end user is discarded and replaced by the fixed value for Code provided in the script.

This example provides a solution using a *Set Variable Data* step type and demonstrates manually unpacking the JSON attribute string. One possible method of doing this is provided in the example. If you prefer, you may instead use the *ConvertFrom-Json* cmdlet similarly to the example for putting approval comments in a metadata field and avoid the manual string manipulation steps.

To create this, add Script Parameters to pull the additional attributes into the script as shown in Figure 189: Additional Attribute Update Example: Add Parameters

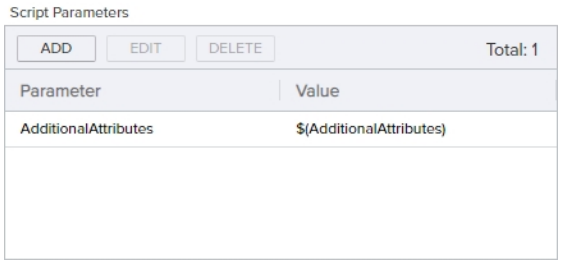


Figure 189: Additional Attribute Update Example: Add Parameters

In the Insert PowerShell Script field, enter a script similar to the following:

```
# Declare your parameters at the beginning
param(
   [string]$AdditionalAttributes
)

# Trim brackets off incoming attribute string
$TrimmedAttributes = $AdditionalAttributes.Substring(1,$AdditionalAttributes.Length-2)

# Replace commas bracketed by quotes in attribute string with a temporary string to facilitate split-
ting (assumes no incoming values contain temp string)
$TempString = "`"######`""
$CleanAttributes = $TrimmedAttributes -replace "`",`"", $TempString

# Split the incoming attribute string into its component values at the temporary string
$SplitAttributes = $CleanAttributes.Split('######')

# Split the incoming attribute string key/value pairs
foreach($attribute in $SplitAttributes){
   $attributeComponents = $attribute.Trim() -split ":"
   $attributeComponents
   Switch($attributeComponents[0].Trim()){
      '"Department"' {$Department = $attributeComponents[1].Substring(1,$attributeComponents
[1].Length-2)}
      '"Code"' {$Code = $attributeComponents[1].Substring(1,$attributeComponents[1].Length-2)}
   }
}

# Initialize a hashtable
$UpdatedAttributes = @{}

# Load original attributes in UpdatedAttributes for the else case
if(![string]::IsNullOrWhiteSpace($Code)) {
```

```
    $UpdatedAttributes['Code'] = $Code
}
if(![string]::IsNullOrWhiteSpace($Department)) {
    $UpdatedAttributes['Department'] = $Department
}

# If the value of Department is "Accounting", then the value of Code must be "G5N145"; override
submitted value--if any--and use fixed value
if($UpdatedAttributes['Department'] -eq "Accounting") {
    $UpdatedAttributes['Code'] = "G5N145"
}

# Return the updated attributes to the workflow in the original parameter as a hashtable
$result = @{ "AdditionalAttributes" = $UpdatedAttributes }
return $result
```

The updated attributes will be submitted to the CA as part of the enrollment package and can be viewed in the workflow instance (see Viewing a Workflow Instance on page 316).

Use Custom PowerShell

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append additional data to it using PowerShell.

Important: Before adding the Use Custom PowerShell step to your workflow, you must upload the script for the step into the Keyfactor Command database using the *POST /Extension/Scripts* API endpoint (see Extensions Scripts on page 1704 and POST Extensions Scripts on page 1709). If you open the workflow definition for editing before adding the script, it will not appear as an available script in the configuration dropdown.

- **Script Parameters**: Add any parameters you will use to pass data into your script. These can contain static values or tokens (see Table 11: Tokens for Workflow Definitions). To add a parameter:
    1. In the Script Parameters section, click **Add**.
    2. In the Add/Edit Parameter dialog, enter a name for the parameter in the **Parameter** field. In the **Value** field, enter either a static value to be passed into the PowerShell script or select from the available tokens to pass the token value into the PowerShell in your parameter.
    3. Click **Save** to save your parameter.

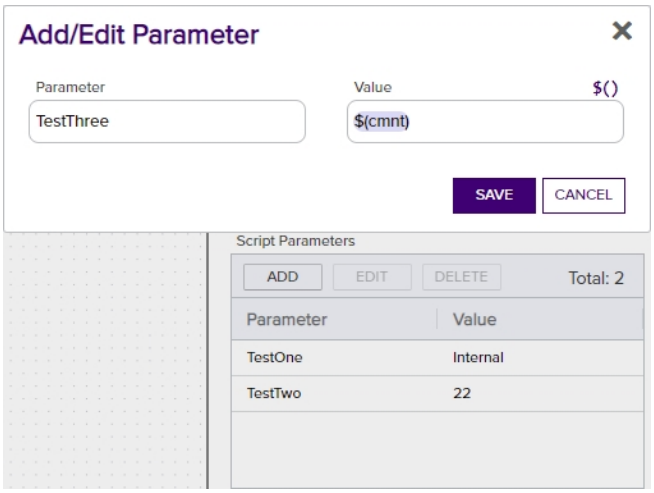


Figure 190: Add Parameters for PowerShell

- **PowerShell Script Name**: The script contents are stored in the Keyfactor Command database. After uploading your script to the database, select a script from the dropdown on the step. All scripts in the database that have been configured for the workflow will be available for selection. See Extensions Scripts on page 1704 for information on adding scripts to the database. The file must be in JSON-escaped format and have an extension of .ps1. The script should use the same input and output method for parameters as described for the Set Variable Data step type (see Set Variable Data on page 275).

  Tip: A sample PowerShell script, **CustomPowershellExample.ps1**, is provided in the workflow directory (default: *C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow*).

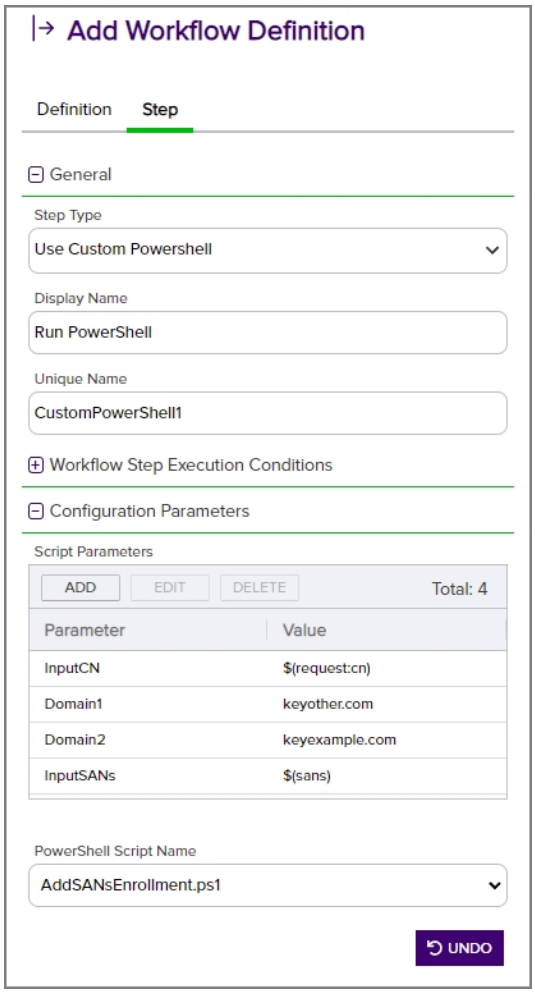


Figure 191: Step Configuration for a Custom PowerShell Workflow Definition Step

Use Custom PowerShell for Enrollment

The following example takes the common name entered during an enrollment and evaluates it to determine whether the domain suffix ends with “keyexample.com”. If it does, the script does a DNS lookup of the full CN to find the IPv4 address for that name and, if found, adds that value as a SAN to the request. Two additional SANs are added to the request by removing the “keyexample.com” domain suffix and instead appending the domain suffixes provided in the Domain1 and Domain2 parameters (e.g. mycert.keyother.com and mycert.keyother2.com). If the CN does not have a domain suffix ending with “keyexample.com”, the PowerShell script does nothing.

This step needs to be a *Use Custom PowerShell* step rather than a *Set Variable Data* step because it calls a PowerShell command (*Resolve-DnsName*) that exists outside the confines of Keyfactor Command.

To do this, first use the *POST /Extension/Scripts* API endpoint (see POST Extensions Scripts on page 1709) to import your PowerShell script into the Keyfactor Command database before beginning to edit your workflow. Your script file prior to import should contain content similar to the following:

```
# Declare your parameters at the beginning ($InputCN, $Domain1, $Domain2, and $InputSANs)
param(
   [string]$InputCN,
   [string]$Domain1,
   [int]$Domain2,
   [string]$InputSANs
)

# Split the incoming SANs string into its component values
$SplitSANs = $InputSANs.Split(',')

# Initialize variables for the two types of SANs we're handling
$DnsSans = @()
$IpSans = @()

# Add the incoming SANs to the correct list (assumes only IPv4 addresses or DNS SANs will be
encountered)
foreach($san in $SplitSANs){
   $sanComponents = $san.Trim() -split ":"
   Switch ($sanComponents[0].Trim()){
      "DnsName" {$DnsSans += ,$sanComponents[1].Trim()}
      "IPAddress" {$IpSans += $sanComponents[1].Trim()}
   }
}

# Check to see if the incoming CN ends with keyexample.com and, if so, add some SANs.
$Suffix = "keyexample.com"

if ($InputCN.EndsWith($Suffix))
{
   # Load just the portion of the CN without the domain name into a variable.
   $CNName = $InputCN.SubString(0,$InputCN.Length - $Suffix.Length)

   # Do a lookup on the requested CN to find its IPv4 address.
   $IPResult = Resolve-DnsName -Name $InputCN -Type A -ErrorAction SilentlyContinue

   # If an address is found, add that address as a SAN.
   # Also add SANs built with the contents of Domain1, Domain2, and the leading part of the CN
   # (e.g. mycert.my-first-other-domain.com and mycert.my-second-other-domain.com).
```

```
   if ($IPResult -ne $null)
   {
      $SAN1 = $IPResult.IPAddress
      $SAN2 = $CNName + $Domain1
      $SAN3 = $CNName + $Domain2
      $DnsSans += ,$SAN2
      $DnsSans += ,$SAN3
      $IpSans += ,$SAN1
   # If an IP address is not found, add only the SANs featuring Domain1 and Domain2.
   else {
      $SAN2 = $CNName + $Domain1
      $SAN3 = $CNName + $Domain2
      $DnsSans += ,$SAN2
      $DnsSans += ,$SAN3
   }
}

# Load the resulting IPv4 and DNS SANs into the SANS variable
$UpdatedSANs = @{}

if(![string]::IsNullOrWhiteSpace($DnsSans)) {
   $UpdatedSANs['dns'] = $DnsSans
}

if(![string]::IsNullOrWhiteSpace($IpSans)) {
   $UpdatedSANs['ip4'] = $IpSans
}

# Return the updated SANs to the workflow as a hashtable (case matters in the return value name
"SANs" in order
# to reload the results back into the SANs token)
$result = @{ "SANs" = $UpdatedSANs; }
return $result
```

Once your PowerShell script has been imported into the Keyfactor Command database, you may begin creating your workflow. To create the Use Custom PowerShell step, add *Script Parameters* to pull the CN and SANs into the script as shown in Metadata Update Example: Add Parameters on page 262, and add to static values to pass in your two additional domain names.

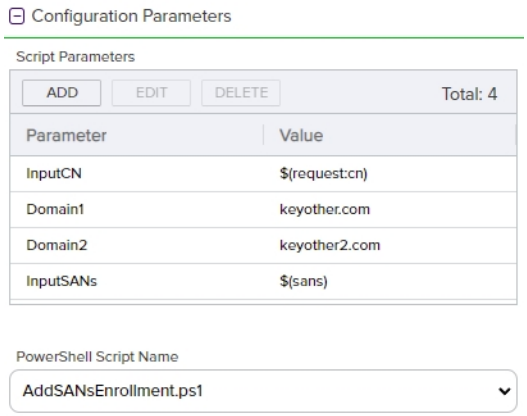


Figure 192: Update SANs Example: Add Parameters

In the *PowerShell Script Name* field dropdown, select the script you uploaded to the database.

Your enrollment will complete using the updated list of SANs, including any SANs you added manually on the PFX enrollment page or in the CSR. You may reference the updated SANs using the standard SANs token ($(sans)) in subsequent steps in your workflow and may view the complete SAN list wherever the SANs are available for viewing within Keyfactor Command.

Note: If you're using a Microsoft CA, in order to add SANs in the workflow you will need to do one of the following:

- Include an Update Certificate Request Subject\SANs step in your workflow (see Update Certificate Request Subject\SANs for Microsoft CAs on page 288). This is Keyfactor's preferred solution for workflow due to the limited risk profile.
- Use Keyfactor's SAN Attribute Policy Handler (see Installing the Keyfactor CA Policy Module Handlers on page 2846). This opens security risks as well, which can be mitigated, however, this is not Keyfactor's preferred solution for workflow.
- Configure your CA to support the addition of SANs outside the initial request (enable the EDITF_ATTRIBUTESUBJECTALTNAME2 flag). Keyfactor does not recommend this solution due to the inherent security risks.

Use Custom PowerShell and Require Approval

The following example takes the approval comment entered when a certificate is enrolled or the approval or denial comment entered when a certificate is revoked using a require approval step and stores the comment in a metadata field. There will be no certificate to associate the metadata field with for an enrollment request that is denied. Normally, approval and denial comments are discarded after a workflow instance is complete, so this allows the comment to be retained.

This example uses a *Use Custom PowerShell* step **after** the *Require Approval* step(s) in the workflow. The Use Custom PowerShell step needs to come after the Require Approval step so that it can include comments gathered from the approve/deny requests.

To do this, first use the *POST /Extension/Scripts* API endpoint (see POST Extensions Scripts on page 1709) to import your PowerShell script into the Keyfactor Command database before beginning to edit your workflow. Your script file prior to import should contain content similar to the following:

```
# Declare your parameters at the beginning
param(
   [string]$ApprovalComment,
   [string]$SignalComment,
   [string]$Metadata
)

# Initialize a hashtable to contain your metadata fields and populate it
$UpdatedMetadata = @{}
$jsonobject = $Metadata | ConvertFrom-Json
foreach( $property in $jsonobject.PSObject.Properties )
{
   $UpdatedMetadata[$property.Name] = $property.Value
}

# Append your signal comment(s) to any existing comment in the ApprovalComment metadata field
if([string]::IsNullOrWhiteSpace($ApprovalComment)) {
   $UpdatedMetadata['ApprovalComment'] = $SignalComment
}else {
   $UpdatedMetadata['ApprovalComment'] = $ApprovalComment + ", " + $SignalComment
}

# Return the updated metadata fields, including ApprovalComment, to the workflow in the original parameter as a hashtable
$result = @{ "ApprovalComment" = $UpdatedMetadata }
return $result
```

Once your PowerShell script has been imported into the Keyfactor Command database, you may begin creating your workflow. To create the Use Custom PowerShell step, add *Script Parameters* to pull any approval comments and the metadata field you're planning to store them in (in this example, a field called ApprovalComments) into the script, along with the metadata bucket to include any remaining metadata values, as shown in Figure 193: Approval Comment Update Example: Add Parameters.

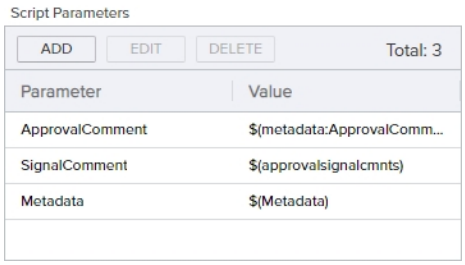


Figure 193: Approval Comment Update Example: Add Parameters

In the *PowerShell Script Name* field dropdown, select the script you uploaded to the database.

The resulting comment will look something like:

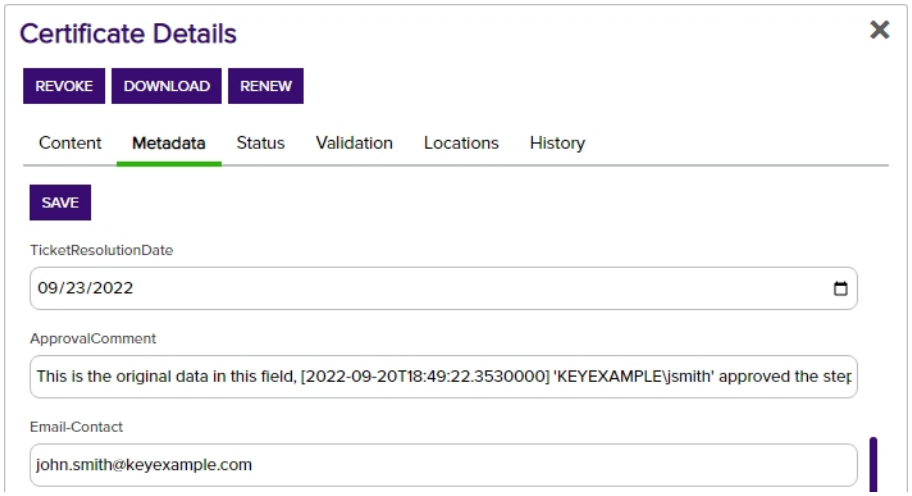


Figure 194: Approval Comment Update Example: Results

If the workflow requires multiple approvals or has multiple require approval steps, all the approval comments entered in the given workflow instance prior to the PowerShell step will be added to the metadata field. If you expect to have multiple comments, you may prefer to use a big text field rather than the string type fields shown here.

Note: If your PowerShell script takes a long time to execute, the step may time out and the workflow instance fail. The default timeout is 60 seconds and is configurable with the *Workflow Step Run Timeout* application setting (see Application Settings: Workflow Tab on page 621).

Update Certificate Request Subject\SANs for Microsoft CAs

This step is used to create a new signed CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) that modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types (see Set Variable Data on page 275 and Use Custom

PowerShell on page 281) or a custom step type. This step is used for both PFX enrollment and CSR enrollment, since both use a CSR that is generated at the start of the workflow. A Microsoft CA will not accept a CSR for enrollment if the subject has been modified and will only accept a CSR for enrollment with modified SANs if the EDITF_ATTRIBUTESUBJECTALTNAME2 flag has been enabled on the CA—a security risk Keyfactor does not recommend. EJBCA doesn't support enroll on behalf of (EOBO), so this step type does not apply to EJBCA CAs. EJBCA is able to handle subject and SAN changes without the need for this type of step based on end entity profile constraints.

- **Enrollment Agent Certificate**: Click **Browse** to search for the desired base-64 encoded PKCS#12 (.PFX) enrollment agent certificate with private key to sign the CSR. This can be either a user certificate or a computer certificate and must have a Certificate Request Agent EKU.
- **Set Private Key Password**: The password for the enrollment agent certificate. Click **Set Private Key Password** to open the *Private Key Password* dialog. Choose the *No Value* checkbox to not assign a password, or choose from Load From Keyfactor Secrets or Load From PAM Provider.

  A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

  > ⚠ **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

  Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

  Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

  ### CyberArk

  Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

  - Safe—The name of the safe the credential resides in.
  - Object—The name of the username or password object in the safe.
  - Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

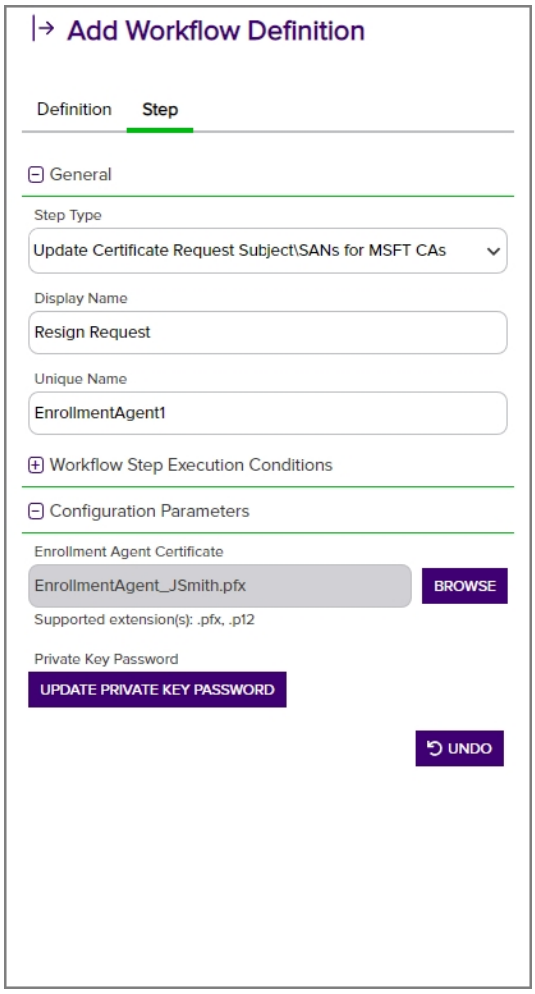


Figure 195: Update Certificate Request Subject\SANs for Microsoft CAs Workflow Definition Step

Update Certificate Request Subject\SANS for MSFT CAs

The following example uses PowerShell to take the distinguished name (subject) and SANs entered during an enrollment along with two static domain names and evaluates the domain name of the common name in the subject to determine whether the domain suffix ends with the "original" domain name provided in the static value ("keyexample.com"). If it does, the script replaces the domain name in the subject with the value provided by the "new" static value and adds a SAN with CN prefix and the new domain name (e.g. CN=mycert.keyexample.com becomes CN=mycert.keyother.com and a SAN is added for mycert.keyother.com). If the CN does not have a domain suffix ending with "keyexample.com", the PowerShell script does nothing. Here we use a *Set Variable Data* step (see Set Variable Data on page 275) since no functions need to be called outside the confines of Keyfactor Command, though you could use a *Custom PowerShell Script* step instead. Then an Update Certificate Subject\SANs for Microsoft CAs step is used to re-sign the request before it is submitted to the CA.

To create this, add Script Parameters to pull the DN and SANs into the script as shown in Metadata Update Example: Add Parameters on page 262 and add two static values to pass in your two domain names.

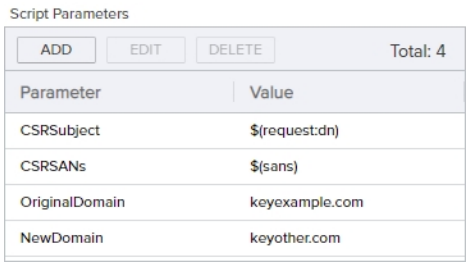


Figure 196: Update SANs and Subject Example: Add Parameters

In the *Insert PowerShell Script* field, enter a script similar to the following:

```
# Declare your parameters at the beginning
param(
   [string]$CSRSubject,
   [string]$CSRSANs,
   [string]$OriginalDomain,
   [string]$NewDomain
)

# Split the incoming SANs string into its component values
$SplitSANs = $CSRSANs.Split(',')

# Initialize variables for the two types of SANs we're handling
$DnsSANs = @()
$IpSANs = @()

# Add the incoming SANs to the correct list (assumes only IPv4 addresses or DNS SANs will be
encountered)
foreach($san in $SplitSANs){
   $sanComponents = $san.Trim() -split ":"
   Switch ($sanComponents[0].Trim()){
      "DnsName" {$DnsSANs += ,$sanComponents[1].Trim()}
      "IPAddress" {$IpSANs += $sanComponents[1].Trim()}
   }
}

# Load original SANs in UpdatedSANs for the else case
$UpdatedSANs = @{}

if(![string]::IsNullOrWhiteSpace($DnsSANs)) {
```

```
   $UpdatedSANs['dns'] = $DnsSANs
}

if(![string]::IsNullOrWhiteSpace($IpSANs)) {
   $UpdatedSANs['ip4'] = $IpSANs
}

# Load original subject in NewSubject for the else case
$NewSubject = $CSRSubject

# Replace escaped commas in the subject temporarily with a string to facilitate splitting
$TempString = "######"
$CleanSubject = $CSRSubject -replace "\\,", $TempString

# Split the incoming Subject string into its component values
$SplitSubject = $CleanSubject.Split(',')

# Initialize variables for the components of the subject
$SubjectCN = @()
$SubjectO = @()
$SubjectOU = @()
$SubjectL = @()
$SubjectST = @()
$SubjectC = @()
$SubjectE = @()

# Load subject values
foreach($element in $SplitSubject){
   $SplitElement = $element.Split('=')
   Switch($SplitElement[0]){
      "CN" {$SubjectCN = $SplitElement[1]}
      "O" {$SubjectO = $SplitElement[1]}
      "OU" {$SubjectOU = $SplitElement[1]}
      "E" {$SubjectE = $SplitElement[1]}
      "L"{$SubjectL = $SplitElement[1]}
      "ST"{$SubjectST = $SplitElement[1]}
      "C"{$SubjectC = $SplitElement[1]}
   }
}

# Check to see if the incoming CN ends with $OriginalDomain and, if so, add it as a SAN with $NewDo-
main and update the Subject with $NewDomain (assumes non-null CN)
if ($SubjectCN.EndsWith($OriginalDomain))
```

```
{
   # Load just the portion of the CN without the domain name into a variable.
   $CNName = $SubjectCN.SubString(0,$SubjectCN.Length - ($OriginalDomain.Length + 1)) # +1 to account
for the '.'

   # Build new DNS SAN
   $NewSAN = $CNName + "." + $NewDomain

   # Add new SAN to DNS SANs
   $DnsSans += ,$NewSAN

   # Build new Subject with $NewDomain
   $NewSubject = "";

   if(![string]::IsNullOrWhiteSpace($SubjectCN)){
      $NewSubject += "CN=" + $CNName + "." + $NewDomain + ","
   }

   if(![string]::IsNullOrWhiteSpace($SubjectO)){
      $NewSubject += "O=" + $SubjectO + ","
   }

   if(![string]::IsNullOrWhiteSpace($SubjectOU)){
      $NewSubject += "OU=" + $SubjectOU + ","
   }

   if(![string]::IsNullOrWhiteSpace($SubjectL)){
      $NewSubject += "L=" + $SubjectL + ","
   }

   if(![string]::IsNullOrWhiteSpace($SubjectST)){
      $NewSubject += "ST=" + $SubjectST + ","
   }

   if(![string]::IsNullOrWhiteSpace($SubjectC)){
      $NewSubject += "C=" + $SubjectC + ","
   }

   if(![string]::IsNullOrWhiteSpace($SubjectE)){
      $NewSubject += "E=" + $SubjectE + ","
   }

   $NewSubject = $NewSubject.Remove($NewSubject.Length - 1) # remove the last ','
```

```
    # Replace temporary string with escaped commas in Subject
    $NewSubject = $NewSubject -replace $TempString, "\,"

    # Load the resulting IPv4 and updated DNS SANs into the SANs variable
    $UpdatedSANs = @{}

    if(![string]::IsNullOrWhiteSpace($DnsSANs)) {
       $UpdatedSANs['dns'] = $DnsSANs
    }

    if(![string]::IsNullOrWhiteSpace($IpSANs)) {
       $UpdatedSANs['ip4'] = $IpSANs
    }
}

# Return the updated subject and SANs as NewSubject and NewSANs to the workflow as a hashtable
$result = @{ "Subject" = $NewSubject; "SANs" = $UpdatedSANs }
return $result
```

Add an Update Certificate Request Subject\SANs for Microsoft CAs step at a point in the workflow **after** your PowerShell step to allow the request to be re-signed before it is submitted to the Microsoft CA for enrollment.

Your enrollment will complete using the updated list of SANs, including any SANs you added manually on the PFX enrollment page or in the CSR, and the updated subject. You may reference the updated SANs using the standard SANs token ($(sans)) and updated subject using the standard DN token ($(request:dn)) in subsequent steps in your workflow and may view the subject and complete SAN list wherever the subject and SANs are available for viewing within Keyfactor Command.

Windows Enrollment Gateway - Populate from AD

This step is needed for any Keyfactor Windows Enrollment Gateway requests where the incoming template (the template from the client side) is configured to build the subject of the certificate request from Active Directory. It has no configuration parameters.

Substitutable Text Tokens for Workflow

Refer to the following table for a list of the substitutable special text tokens that are available in the dropdown to customize workflow email messages, conditions, and select parameter configuration fields along with a selection of some additional tokens that are not found in the dropdown but which exist in the data bucket (see tip).

Tip: In addition to the tokens in the dropdown, any data in the current data bucket can be referenced by entering an appropriate reference string. For example, to return the CSR for

an enrollment request you can use **$(CSR)**. Refer to the *CurrentStateData* field in the response to the GET /Workflow/Instances/{instanceId} API method for information on all the data found in the current (as opposed to initial) data bucket (see GET Workflow Instances Instance ID on page 2630).

Table 11: Tokens for Workflow Definitions

Variable	Name	Request Type	In Drop-down?	Description
$(AdditionalAttributes)	n/a	Enrollment	No	An array containing the additional enrollment fields, if any, in key value pair format.
$(approvalsignalcmnts)	Workflow Approval or Denial Comment	Certificate Collection, Enrollment and Revocation	Yes	The comment provided when a workflow request that requires approval is approved or denied.
$(CA)	Issuing CA	Certificate Collection, Enrollment and Revocation	Yes	A string containing the Issuing CA logical name and hostname.
$(CACertificate)	n/a	Enrollment	No	Any array of information about the chain certificate(s) for the request, including the certificate.
$(certid)	Request ID	Certificate Collection and Revocation	Yes	The request ID for the certificate as stored in the Keyfactor Command database. This is not the same as the request ID issued by the CA.
$(cmnt)	Revocation Comment	Revocation	Yes	The comment entered at revocation time to explain the revocation.
$(code)	Revocation	Revoc-	Yes	The reason selected at revocation

Variable	Name	Request Type	In Drop-down?	Description
	Reason	ation		time to explain the revocation as a string (e.g. AffiliationChanged).
$(cn)	Common Name	Certificate Collection and Revocation	Yes	The certificate common name.
$(CSR)	n/a	Enrollment	No	The CSR generated for the enrollment.
$(CustomName)	n/a	Enrollment	No	The custom friendly name, if any, set for the certificate on enrollment.
$(dn)	Distinguished Name	Certificate Collection and Revocation	Yes	The certificate distinguished name.
$(effdate)	Revocation Effective Date	Revocation	Yes	Date on which the revocation becomes effective.
$(Format)	n/a	Enrollment	No	The value selected during PFX Enrollment for the format for the certificate. Possible values are: JKS, PFX, Store, Zip
$(IsPFX)	n/a	Enrollment	No	A Boolean indicating whether the certificate request was made using the PFX Enrollment method in Keyfactor Command (true) or not (false).
$(issuerdn)	Issuer DN	Certificate Collection and Revocation	Yes	The distinguished name of the issuer of the certificate.

Variable	Name	Request Type	In Drop-down?	Description
$(KeyRetention)	n/a	Enrollment	No	A Boolean indicating whether the private key for the certificate has been retained in Keyfactor Command (true) or not (false).
$(keysize)	Key Size	Certificate Collection and Revocation	Yes	The key size of the certificate.
$(keytype)	Key Type	Certificate Collection and Revocation	Yes	The key type of the certificate.
$(locations)	Certificate Store Locations	Certificate Collection, Enrollment and Revocation	Yes	The certificate store locations to which the certificate will be deployed following enrollment, for enrollment requests, or in which the certificate is found, for revocation requests.
$(managementjobtime)	n/a	Enrollment	No	The schedule for the management job to add the certificate to certificate stores on issuance. The field, if populated, will have a value of either “Immediate”: true or “ExactlyOnce” with the date and time at which the management job should begin.
$(request:cn)	Requested Common Name	Enrollment	Yes	The common name contained in the certificate request.
$(request:dn)	Requested Distinguished	Enrollment	Yes	The distinguished name contained in the certificate request.

Variable	Name	Request Type	In Drop-down?	Description
	Name			
$(request:keysize)	Request Key Size	Enrollment	Yes	The key size contained in the certificate request.
$(request:keytype)	Request Key Type	Enrollment	Yes	The key type contained in the certificate request.
$(requester)	Requester	Enrollment and Revocation	Yes	The user account that requested the certificate from the CA, in the form *DOMAIN\username*.
$(requester:mail)	Requester's Email	Enrollment and Revocation	Yes	The email address retrieved from Active Directory of the user account that requested the certificate from the CA, if present. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
$(requester:givenname)	Requester's First Name	Enrollment and Revocation	Yes	The first name retrieved from Active Directory of the user account that requested the certificate from the CA, if present. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
$(requester:sn)	Requester's Last Name	Enrollment and Revocation	Yes	The last name retrieved from Active Directory of the user account that requested the certificate from the CA, if present. **Note:** This substitutable special text token is only

Variable	Name	Request Type	In Drop-down?	Description
				supported in environments using Active Directory as an identity provider.
$(requester:displayname)	Requester's Display Name	Enrollment and Revocation	Yes	The display name retrieved from Active Directory of the user account that requested the certificate from the CA, if present. **Note:** This substitutable special text token is only supported in environments using Active Directory as an identity provider.
$(reviewlink)	Review Link	Certificate Collection, Enrollment and Revocation	Yes	Link pointing to the review page in the Management Portal for the workflow instance where the person responsible for providing signal input (e.g. approving the request) can go to review the request and provide the input. **Note:** This option is only useful in workflows that contain a step that requires signal input (e.g. requires approval).
$(RevokeCode)	n/a	Revocation	No	The reason selected at revocation time to explain the revocation as an integer (e.g. 3). See also *$(code)*. For details on the mapping of numeric revocation codes to revocation strings, refer to the *POST /Certificates/Revoke* API endpoint (see POST Certificates Revoke on page 1166).
$(sans)	Subject Alternative	Enrollment	Yes	Subject alternative name(s) contained in the certificate request.

Variable	Name	Request Type	In Drop-down?	Description
	Names			There are four possible sources for the SANs that appear here: • For CSR enrollment, the original SANs included in the CSR. • Any SANs added through the Keyfactor Command Management Portal. For CSR enrollment, these take the place of the SANs in the CSR if the ATTRIBUTESUBJECTALTNAME2 option is enabled on the CA. See CSR Enrollment on page 136. • A SAN matching the CN added automatically during enrollment as a result of setting the RFC 2818 compliance flag in the CA configuration. See Adding or Modifying a CA Record on page 354. For PFX enrollment, the user has the option of editing this entry at enrollment time; entry of something is required. • A SAN matching the CN added automatically by the Keyfactor Command policy module on the CA if the Keyfactor Command RFC 2818 Policy Handler is enabled, if one was not included in the CSR or added manually. See Installing the

Variable	Name	Request Type	In Drop-down?	Description
				Keyfactor CA Policy Module Handlers on page 2846.
$(serial)	Serial Number	Certificate Collection and Revocation	Yes	Certificate serial number.
$(stores)	n/a	Enrollment	No	The certificate store(s) to which the certificate will be delivered on issuance.
$(subdate)	Submission Date	Enrollment and Revocation	Yes	Date the workflow was initiated.
$(template)	Template Name	Certificate Collection and Enrollment	Yes	The short name (often the name with no spaces) of the certificate template used to create the certificate request.
$(thumbprint)	Thumbprint	Certificate Collection and Revocation	Yes	Thumbprint of the certificate.
$(metadata:EmailContact)	EmailContact	Certificate Collection, Enrollment and Revocation	Yes	Example of a custom metadata field. Your custom metadata fields would be referenced similarly (e.g. $(metadata:AppOwnerFirstName) for metadata field AppOwnerFirstName).

Using the Workflow Definitions Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

DisplayName

Complete or partial matches with the name of the workflow definition.

Id

The Keyfactor Command reference GUID for the workflow definition.

IsPublished

The workflow has been published yes/no.

WorkflowType

The type of workflow (enrollment, revocation, certificateenteredcollection, or certificateleftcollection).

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

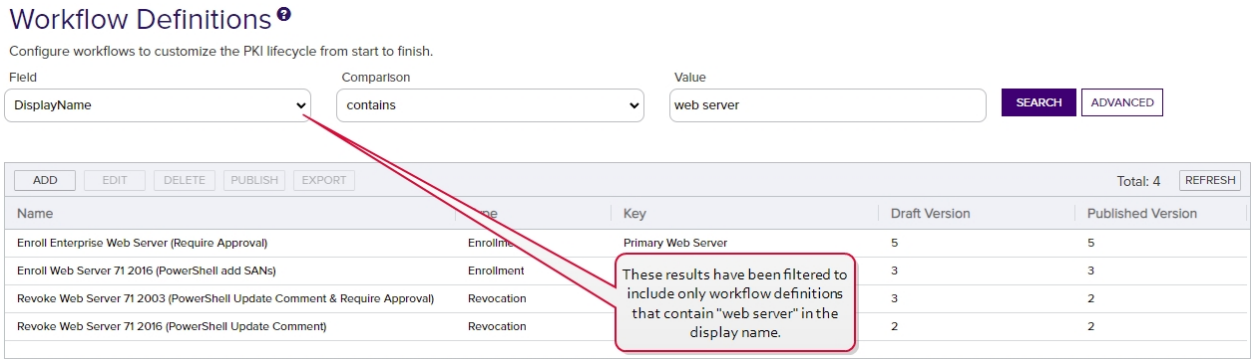


Figure 197: Simple Workflow Definitions Search

The search results can be sorted by clicking on a column header in the results grid for several of the columns. Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may be adjusted by click-holding and dragging the line separating two column headers.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you

click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

2.1.7.2 Workflow Instances

Whether you've chosen to create custom workflow definitions (see Workflow Definitions on page 230) or are relying on the built-in global workflow definitions, all certificate enrollments, renewals, and revocations go through workflow and create workflow instances. Certificate collection additions and removals only go through workflows if you create custom workflows for these actions, as there aren't built-in global workflows for these functions. The workflow instance is the combination of the certificate action and the workflow definition for that action as defined at the time that action took place.

Example: You have a custom enrollment workflow definition for the EnterpriseWebServer template. It contains a couple of steps including RequireApproval, which requires approval from at least two PKI admins before a certificate with this template may be issued. The workflow definition has been edited and published a few times and is now at version 3. John enrolls for a certificate using the Management Portal PFX Enrollment option and selects this template. When the enrollment completes, he receives a message indicating that the request is awaiting approval.

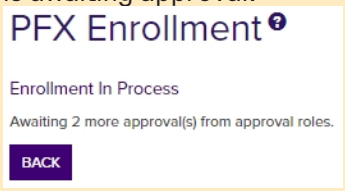


Figure 198: PFX Enrollment Complete for a Template Requiring Approval via Workflow

A workflow instance has now been created for his request. Users with appropriate permissions can view the instance in *Workflow Instances*.

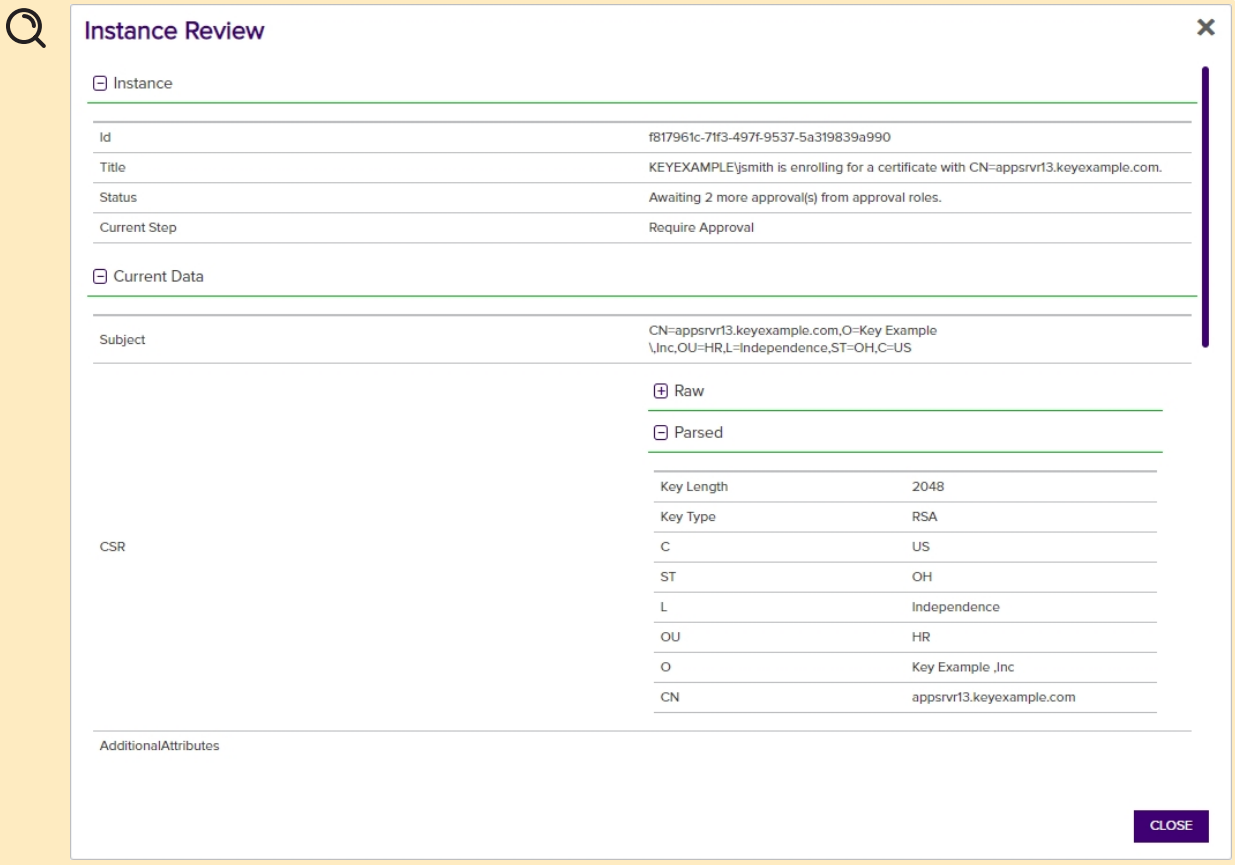


Figure 199: View Workflow Instance for a PFX Enrollment

Users with permissions to approve the request can do so through their *My Workflows* page and the *Assigned to Me* tab (see My Workflows on page 324).

After John completes his enrollment and before it is approved, an administrator makes a change to the workflow for the EnterpriseWebServer template and publishes the new version. The current workflow is now at version 4. However, John's request remains outstanding and valid with version 3 of the workflow. Any change made for version 4 of the template will not be reflected in John's request.

The only circumstance under which John's request might complete using version 4 of the workflow definition would be:

- If the administrator observed the suspended workflow (suspended because it is awaiting approvals), knew there was a new version of the workflow, and pro-actively restarted the workflow instance. A workflow instance restarted from a suspended state will always restart (from the beginning) with the currently active version of the workflow definition.
- If the administrator observed the suspended workflow, stopped the workflow knowing it should not be allowed to complete with the workflow definition it was submitted with, made

a further update to the workflow definition, and then restarted the workflow with the newly updated version of the workflow definition. One common reason to stop and restart rather than just restarting would be to allow time to make changes to the workflow.

- If the original request failed for some reason (e.g. the CA was not responding when the final approval was received and the request was submitted to the CA) and the administrator chose to restart the failed request with the currently active version of the workflow definition (the default) rather than the original version of the workflow after resolving the reason for the failure.

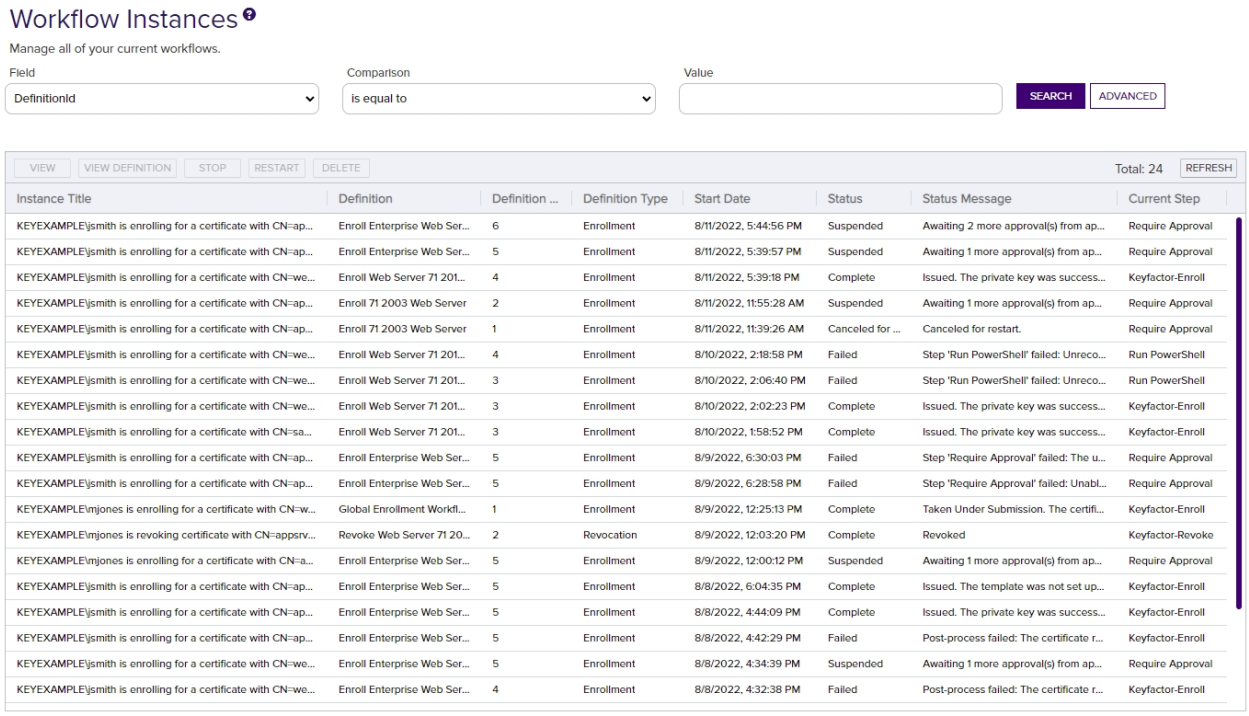


Figure 200: Workflow Instances

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Workflow Instances Operations

A workflow instance is created for *every* certificate enrollment, renewal, or revocation request you make through the Keyfactor Command Management Portal. The addition and removal of certificates from certificate collections can be configured to flow through workflow as well, but these create workflow instances only if configured. If a given request is made using a workflow definition (see

Workflow Definitions on page 230) that has been configured with steps to require approvals for the request, run a PowerShell script, or make an API request as part of the request flow, you may find yourself on the Workflow Instances page needing to manage the instances.

Workflow instance operations include:

- Viewing a workflow instance to review details of the instance
- Viewing the workflow definition as configured for the particular workflow instance to understand the configuration at the time the instance was initiated
- Stopping a workflow instance
- Restarting a workflow instance after correcting a failure (e.g. the CA was not responding on an enrollment) or to introduce a different workflow definition
- Deleting workflow instances to clean house

Depending on the page from which the workflow instance is viewed from, different data will be shown.

Instance Data Details

Instance Section Data

Name	Description
Id	A GUID indicating the Keyfactor Command reference ID for the instance.

Title	A description for the action taking place in the step. For example:
	`"KEYEXAMPLE\jsmith is enrolling for a certificate with CN=appsrvr12.keyexample.com."`
	Or:
	`"KEYEXAMPLE\mjones is revoking certificate with CN=appsrvr14.keyexample.com."`

Status	The current status message of the workflow instance. For example, for an enrollment that **succeeded**, the status message might be: `Issued. The private key was successfully retained.` For a workflow **suspended and awaiting approval**, the status message might be:

Name	Description
	``` Awaiting 2 more approval(s) from approval roles. ``` For an **enrollment** that could not be submitted because a regular expression rule was not met, the status message might be something like: ``` Pre-process failed: Invalid ST provided: Value must be one of California, Washington, Texas, New York, Illinois or Ohio. ``` For an **enrollment** that failed due to rejection by the CA, the status message might be: ``` The certificate request failed with the reason '[CA reason]' ``` A workflow that failed at a **PowerShell step** might include the PowerShell error in the status message: ``` Step 'Run PowerShell' failed: At line:5 char:19 + [datetime]$Date + ~ Missing ')' in function parameter list. At line:7 char:1 +) + ~ Unexpected token ')' in expression or statement. ```
Current Step	The display name defined for the workflow instance step at which the instance has paused or stopped. For a successfully completed enrollment or revocation workflow, this will be either *Keyfactor-Revoke* or *Keyfactor-Enroll*. For a suspended workflow, this will be the custom step that is awaiting user input to continue the workflow. For a failed workflow, this will be the step at which the workflow failed.

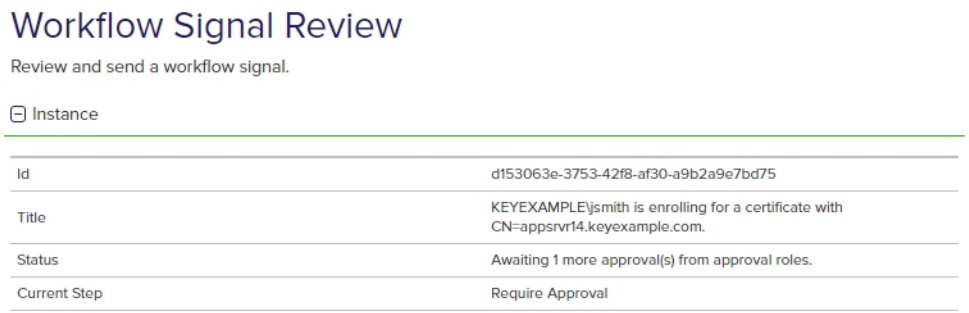


Figure 201: Workflow Instance Review

Current Data Section

The data included in this section will vary depending on the request type, the status of the request, and the configuration of the workflow.

Enrollment

Name	Description	
Subject	The distinguished name of the certificate.	
CSR:Raw	The unparsed version of the certificate signing request generated for the certificate request.	
CSR: Parsed	The parsed version of the certificate signing request generated for the certificate request. The CSR may include: • Key Length The desired key size for the certificate. • Key Type The desired key encryption for the certificate. • C The country (two characters) of the certificate. • ST The state or province of the certificate. • L The city or locality of the certificate. • OU The organizational unit of the certificate. • O The organization of the certificate. • E The email address of the certificate.	

Name	Description	
	• CN The common name of the certificate. • DNS Name A SAN value containing a DNS name. • IP Address A SAN value containing an IP v4 or IP v6 address. • RFC822 Name A SAN value containing an email address. • Other name:Principal A SAN value containing a user principal name (UPN).	
Additional Attributes	Values for any custom enrollment fields set on the certificate template to supply custom request attributes to the CA during the enrollment process.	
CA Certificate	The certificate information returned from the CA for the certificate that is being requested, including: • CA Certificate ID The ID assigned to the certificate by the CA. • CA Request ID The ID assigned to the certificate request by the CA. • Status The numeric status for the certificate as returned by the CA. • Certificate Template The certificate template used to issue the certificate. • Revocation Date The revocation date for the certificate as returned by the CA, if applicable (generally not for newly enrolled certificates). • Revocation Reason The revocation reason for the certificate as returned by the CA, if applicable (generally not for newly enrolled certificates). • Archived Key A flag indicating whether the certificate is configured for key archival on the CA (true) or not (false).  **Note:** This field is populated only after the certificate has been issued by the CA.	

Name	Description	
CA Certificate Data: Raw	The certificate as returned by the CA in base-64 encoded binary format.	
CA Certificate Data: Parsed	• Issued DN The distinguished name of the certificate. • Issuer DN The distinguished name of the issuer. • Thumbprint The thumbprint of the certificate. • Not After The date, in UTC, on which the certificate expires. • Not Before The date, in UTC, on which the certificate was issued by the certificate authority. • Metadata The metadata fields populated for the certificate.	
CA Certificate Request	The certificate request information returned from the CA for the certificate that is being requested, including: • CA Request ID The ID assigned to the certificate request by the CA. • CSR The certificate signing request for the certificate request as returned by the CA. • Status The status for the certificate as returned by the CA. • Requester Name The requester name on the certificate request as returned by the CA. **Note:** This field is populated only if the certificate request fails at the CA level or requires manager approval at the CA level.	
Certificate Authority	The certificate authority that will be used to enroll against in *host-name\logical name* format.	
Custom Name	A custom friendly name for the certificate, if entered at enrollment.	

Name	Description	
Disposition Message	A message about the certificate request. **Note:** This field is populated only after the certificate request has been submitted to the CA.	
Format	The desired output format for the certificate. A value of STORE indicates that the certificate is intended to be delivered into one or more certificate stores.	
Include Chain	A flag indicating whether to include the certificate chain in the enrollment response (true) or not (false).	
Initiating User Name	The name of the user who initiated the workflow in DOMAIN\username format.	
Is PFX	A flag indicating whether the certificate enrollment type that initiated the workflow instance was PFX (true) or CSR (false).	
Issuer DN	The distinguished name of the issuer.	
Key Retention	A flag indicating whether the private key for the certificate resulting from the enrollment will be retained in Keyfactor Command (true) or not (false).	
Key Status	A numeric value indicating the status of the private key retention for the certificate within Keyfactor Command. Possible values are: • 0—Unknown • 1—Saved • 2—Expected • 3—NoRetention • 4—Failure • 5—Temporary	
Keyfactor Id	The Keyfactor Command reference ID for the certificate.	
Management Job Time	The schedule for the management job to add the certificate to any certificate store(s).	
Metadata	Values for the metadata fields that will be associated with the certificate once it is in Keyfactor Command.	

Name	Description	
PFX Password Secret Instance Id	The Keyfactor Command reference ID for the PFX password used to secure the PFX file on download.	
Private Key Converter	An internally used Keyfactor Command field.	
Renewal Certificate	Certificate Id - The Keyfactor Command reference ID of the certificate this certificate replaces on a renewal.	
Renewal Certificate Data: Raw	The certificate that this certificate replaces on a renewal as returned by the CA in base-64 encoded binary format.	
Renewal Certificate Data: Parsed	The certificate details for the certificate that this certificate replaces on a renewal, including: • Issued DN The distinguished name of the certificate. • Issuer DN The distinguished name of the issuer. • Thumbprint The thumbprint of the certificate. • Not After The date, in UTC, on which the certificate expires. • Not Before The date, in UTC, on which the certificate was issued by the certificate authority. • Metadata The metadata fields populated for the certificate.	

SANs: Type	The subject alternate names defined for the certificate. Possible types that can be entered within Keyfactor Command are DNS Name, IPv4 Address, IPv6 Address, User Principal Name, and Email. Within each type is a list of entries for that type shown with a key name of the entry number and the actual value. For example, if you had two DNS SANs, the DNS Name section would look something like:
	`Entry 1: myfirstsan.keyexample.com`
	`Entry 2: mysecondsan.keyexample.com`	

Name	Description	
Serial Number	The serial number of the certificate.	
Stores	The certificate stores to which the certificate should be distributed, if applicable.	
Template	The template that was used when requesting the certificate.	
Thumbprint	The thumbprint of the certificate.	
(Custom)	Optional user-generated custom fields returning response data from PowerShell scripts or REST requests. These will be sorted alphabetically in among the other fields on the page.	

Revocation

For a revocation, this section may include:

Name	Description
Certificate Authority	The certificate authority that issued the certificate.
Certificate Id	The Keyfactor Command reference ID for the certificate being revoked.
Comment	A freeform reason or comment to explain why the certificate is being revoked.
Delegate	A flag indicating whether delegation was enabled for the certificate authority that issued the certificate at the time revocation was requested (true) or not (false). For more information, see Authorization Methods Tab on page 367.
Effective Date	The date and time when the certificate will be revoked.
Initiating User Name	The name of the user who initiated the workflow in DOMAIN\\username format.
Operation Start	The time at which the revocation workflow was initiated.
RevokeCode	The specific reason that the certificate is being revoked. Possible values are: • -1—Remove from Hold • 0—Unspecified • 1—Key Compromised • 2—CA Compromised • 3—Affiliation Changed

Name	Description
	• 4—Superseded • 5—Cessation of Operation • 6—Certificate Hold • 7—Remove from CRL. Only valid in the case that a cert is already on a CRL in a manner that it can be removed, such as Certificate Hold.
Serial Number	The serial number of the certificate being revoked.
Thumbprint	The thumbprint of the certificate being revoked.
(Custom)	Optional user-generated custom fields returning response data from PowerShell scripts or REST requests. These will be sorted alphabetically in among the other fields on the page.

Certificate Entered Collection and Certificate Left Collection

For a certificate that entered or left a certificate collection, this section generally includes:

Name	Description
Certificate Id	The Keyfactor Command reference ID for the certificate added to or removed from the certificate collection.
Initiating User Name	The name of the user who initiated the workflow—generally *Timer Service* in this case.

Operations

Viewing a Workflow Instance

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Instances > Read

To view a workflow instance:

1. In the Management Portal, browse to *Workflow > Workflow Instances*.
2. On the Workflow Instances page, double-click or click **View** from either the top or right click menu.
3. The Instance Review dialog includes the following information:.

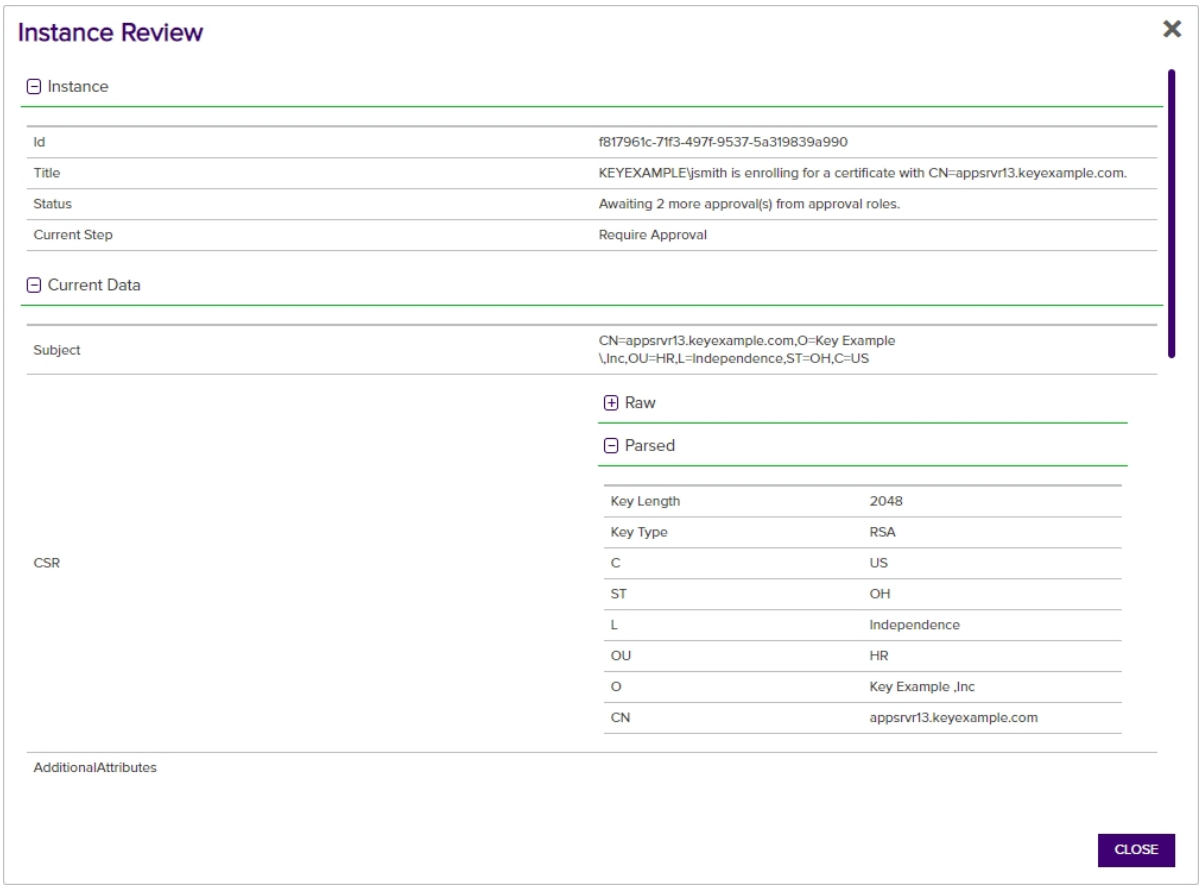


Figure 202: View a Workflow Instance

4. Click **Close** to close the viewer.

Viewing a Workflow Instance Definition

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Instances > Read
Workflows > Definitions > Read

The workflow definition as it existed at the time a particular workflow instance was generated may not necessarily match the current workflow definition. Using the Workflow Definition option on the Workflow Instances page, you can view the workflow definition for the selected instance as it was at the time the instance was initiated using the workflow workspace.

To view a workflow instance definition:

1. In the Management Portal, browse to *Workflow > Workflow Instances*.
2. On the Workflow Instances page, select a workflow instance and click **View Definition** from either the top or right-click menu.
3. A read-only copy of the workflow definition at the time the instance was initiated will open in the workflow definition workspace. For information about using the workflow definition workspace, see Adding, Copying or Modifying a Workflow Definition on page 237.

Stopping a Workflow Instance

If a workflow instance has been initiated in error or with a workflow definition that is not configured correctly, you have the option to stop the workflow instance.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Instances > Read
Workflows > Instances > Manage

To stop a workflow instance:

1. In the Management Portal, browse to *Workflow > Workflow Instances*.
2. On the Workflow Instances page, select a workflow instance and click **Stop** from either the top or right-click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Note: Only workflow instances with a Status of *Suspended* can be stopped.

Restarting a Workflow Instance

If a workflow instance has failed or been stopped to correct an issue, you may restart it to reinitialize the request after correcting whatever issue caused the failure (e.g. a PowerShell script failed or a CA was not responding on enrollment). You may also choose to use restart if a workflow instance was initiated with a workflow definition that had an incorrect definition that can easily be corrected—for example, the definition requires approval from just one user and that user is no longer available. In this case, you can update the definition, republish it, and then restart the workflow with the latest published version.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Definitions > Read
Workflows > Instances > Read
Workflows > Instances > Manage

When you restart a workflow instance, it starts over from the beginning, not from the failure point.

To restart a workflow instance:

1. In the Management Portal, browse to *Workflow > Workflow Instances*.
2. On the Workflow Instances page, select a workflow instance and click **Restart** from either the top or right-click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Note: Only workflow instances with a Status of *Failed* or *Suspended* can be restarted.

After restarting a workflow instance, you can view any differences between the original instance and the newly restarted instance by looking at the audit log record (see Audit Log Operations on page 721) for the workflow instance restart. The Related Entries in the audit log record do not include the original workflow instance that failed since restarting a workflow instance generates a new workflow instance.

Tip: If user John Smith restarts a workflow instance that was originally started by user Martha Jones, the audit log message for this will look something like:

```
"The user 'KEYEXAMPLE\jsmith' restarted workflow instance, 'KEYEXAMPLE\mjones is
enrolling for a certificate with CN=appsrvr12.keyexample.com.'"
```

In a scenario like this, the user listed at the top of the audit log details will be the user who restarted the instance, not the user who originally started the request.

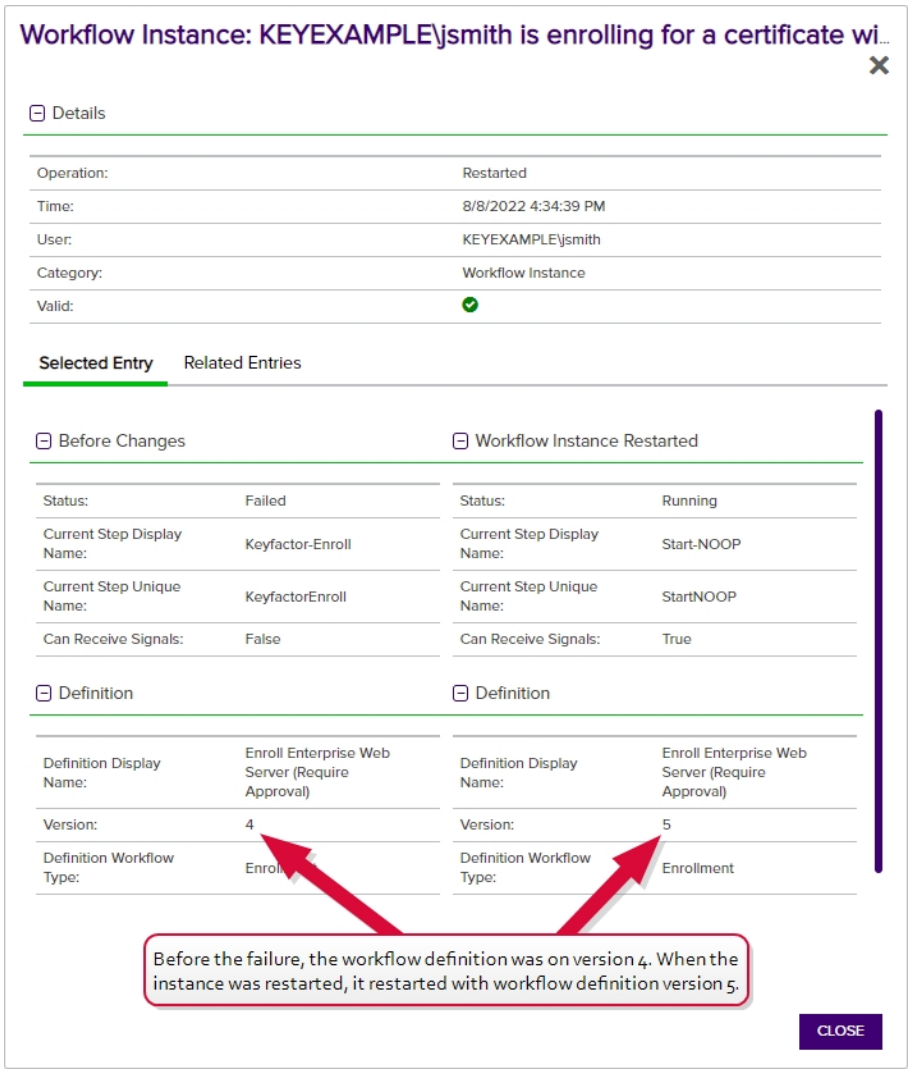


Figure 203: View an Audit Log Entry for a Restarted Workflow Instance

Deleting a Workflow Instance

If a workflow instance has failed, you may wish to remove the failed instance from the grid.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Instances > Read
Workflows > Instances > Manage

To delete a workflow instance:

1. In the Management Portal, browse to *Workflow > Workflow Instances*.
2. On the Workflow Instances page, select a workflow instance and click **Delete** from either the top or right-click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

An audit log entry is created when you delete a workflow instance (see Audit Log on page 716). Instances deleted as the result of system action (e.g. purging old records) are not audited.

Using the Workflow Instances Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

DefinitionId

The Keyfactor Command reference GUID for the workflow *definition*.

Id

The Keyfactor Command reference GUID for the workflow *instance*.

InitiatingUserName

Complete or partial matches with the name of the user who initiated the workflow instance in DOMAIN\username format for Active Directory authentication or username for an identity provider other than Active Directory.

LastModified

The date and time on which an initiated instance was last updated. The instance is updated each time a step in the workflow is completed, when

StartDate

The date and time when an instance was initiated.

Status

Status matches or doesn't match the referenced value. Supported statuses are:

- Unknown
- Running
- Suspended
- Failed
- Complete
- Rejected
- CanceledforRestart

Title

Complete or partial matches with the description for the action taking place in the workflow instance step. The values in the title will vary and generally

signals are received for a step that accepts signals (e.g. a requires approval step), or when an instance is stopped or restarted.

include the user initiating the request and the CN of the certificate or certificate request involved.

WorkflowType

The type of workflow (enrollment, revocation, certificateenteredcollection, or certificateleftcollection).

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

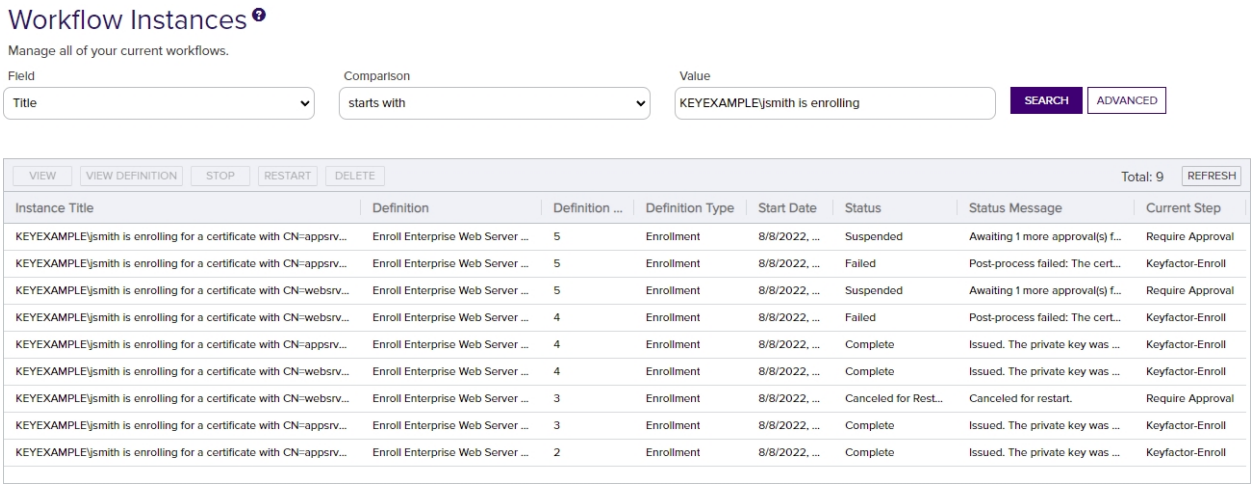


Figure 204: Simple Workflow Instance Search

The search results can be sorted by clicking on a column header in the results grid for several of the columns. Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may be adjusted by click-holding and dragging the line separating two column headers.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template

referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

2.1.7.3 My Workflows

When a workflow is initiated by a certificate enrollment, renewal, revocation request, or automated task (for workflows of types Certificate Entered Collection and Certificate left Collection), that workflow instance may appear in as many as two places:

- If the workflow definition for the instance requires signal input (e.g. approval), every Keyfactor Command user who holds a security role that has been defined in the workflow definition as allowed to send signals to the workflow (see Workflow Definitions on page 230) will see that instance appear on the *Assigned to Me* tab of the My Workflows page. The users can provide signal input (e.g. approve or deny the request) from here. The workflow does not necessarily need to receive signal input from all these users, depending on how many users with this role there are and how many users were required to provide signal input in the workflow definition. Once the workflow instance is complete, it disappears from the *Assigned to Me* tab for all users.
- The user who initiated the workflow (e.g. by beginning a certificate enrollment or revoking a certificate) will see that instance appear on the *Created by Me* tab of the My Workflows page. When the workflow instance is complete, it will still appear on the *Created by Me* tab and be searchable.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Instances > Read OR
Workflows > Instances > Read > Mine

Users with only *Read > Mine* will only be able to see the *Created by Me* or *Assigned to Me* tab, respectively. A user with *Read* will be able to see both tabs.

Example: The enrollment workflow definition for the *EnterpriseWebServer* template requires two approvals from users with the *Enrollment Approvers* security role. There are five users with this role: Anne, Charles, John, Mary, and Sam. Martha enrolls for a certificate using the Keyfactor Command Management Portal PFX Enrollment method and the *EnterpriseWebServer* template.

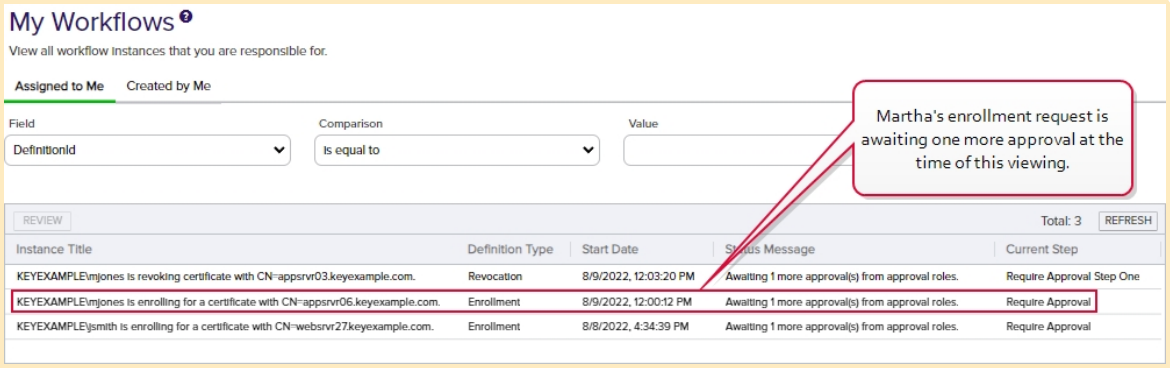


Figure 205: Workflows Assigned to Mary

The new workflow instance appears on the *Assigned to Me* tab of all users with the *Enrollment Approvers* role and on Martha's *Created by Me* tab. Approvers Mary and John approve the instance on their respective *Assigned to Me* tab and the certificate is issued. The workflow instance disappears from the *Assigned to Me* tab for all users. It's still visible on the main *Workflow Instances* page and on Martha's *Created by Me* tab as a completed instance.

Note: A locking conflict may occur if two (or more) users attempt to provide input to a workflow instance (e.g. approve a request) at exactly the same time. If this happens, input from only one of the users will be reflected in the Management Portal, and the workflow instance will not be moved along to the next step if it should have been with input from the two users. The other input is still accepted, however, and there is a scheduled task that runs daily and attempts to continue all suspended workflows that may be eligible to continue but have not done so due to locking conflicts.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Workflows Assigned to Me Operations

Only workflow instances that are in a Suspended state and that the current user has permissions to submit signals for (e.g. approve or deny) appear on the Assigned to Me tab of the My Workflows page. Once the user submits a signal to a workflow instance on this page, it is removed from the page.

Note: If a workflow instance is initiated for a workflow definition that has more than one step requiring input (signals), a user can only provide that input (e.g. approve or deny a require approval request) at the step in the workflow instance where the workflow instance was

suspended pending input. The user cannot jump ahead and provide input for future steps in the workflow that have not yet occurred.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Instances > Read > Mine
Or:
Workflows > Instances > Read

To review a workflow instance and potentially submit a signal for it:

1. In the Management Portal, browse to *Workflow > My Workflows*.
2. On the Assigned to Me tab of the My Workflows page, double-click or click **Review** from either the top or right click menu.
3. On the Workflow Signal Review page, review the information in the instance before submitting a signal for the request. Information on the review page includes:

Name	Description
Id	A GUID indicating the Keyfactor Command reference ID for the instance.

Title	A description for the action taking place in the step. For example:
	`"KEYEXAMPLE\jsmith is enrolling for a certificate with CN=appsrvr12.keyexample.com."`
	Or:
	`"KEYEXAMPLE\mjones is revoking certificate with CN=appsrvr14.keyexample.com."`

Status	The current status message of the workflow instance. For a workflow **suspended and awaiting approval**, the status message might be: `Awaiting 2 more approval(s) from approval roles.`
Current Step	The display name defined for the workflow instance step at which the instance has paused. For a suspended workflow, this will be the custom step that is awaiting user input to continue the workflow.

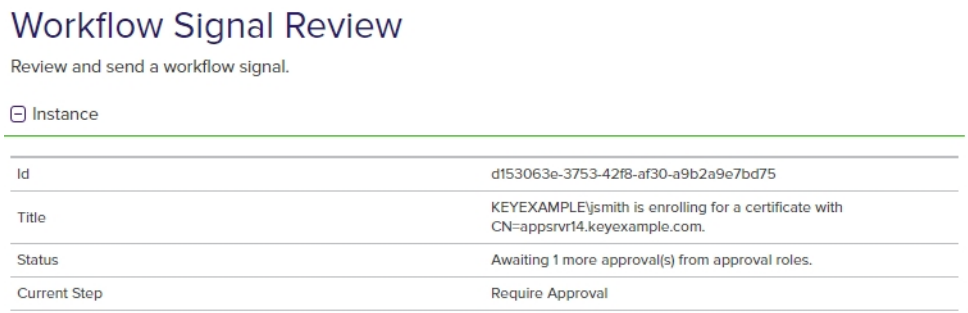


Figure 206: Workflow Instance Review

The data included in this section will vary depending on the request type and the configuration of the workflow.

Enrollment

Name	Description	
Subject	The distinguished name of the certificate.	
CSR:Raw	The unparsed version of the certificate signing request generated for the certificate request.	
CSR: Parsed	The parsed version of the certificate signing request generated for the certificate request. The CSR may include: • Key Length The desired key size for the certificate. • Key Type The desired key encryption for the certificate. • C The country (two characters) of the certificate. • ST The state or province of the certificate. • L The city or locality of the certificate. • OU The organizational unit of the certificate. • O The organization of the certificate. • E	

Name	Description	
	The email address of the certificate. • CN The common name of the certificate. • DNS Name A SAN value containing a DNS name. • IP Address A SAN value containing an IP v4 or IP v6 address. • RFC822 Name A SAN value containing an email address. • Other name:Principal A SAN value containing a user principal name (UPN).	
Additional Attributes	Values for any custom enrollment fields set on the certificate template to supply custom request attributes to the CA during the enrollment process.	
Certificate Authority	The certificate authority that will be used to enroll against in *hostname\logical name* format.	
Custom Name	A custom friendly name for the certificate, if entered at enrollment.	
Format	The desired output format for the certificate. A value of STORE indicates that the certificate is intended to be delivered into one or more certificate stores.	
Include Chain	A flag indicating whether to include the certificate chain in the enrollment response (true) or not (false).	
Initiating User Name	The name of the user who initiated the workflow in DOMAIN\username format.	
Is PFX	A flag indicating whether the certificate enrollment type that initiated the workflow instance was PFX (true) or CSR (false).	
Key Retention	A flag indicating whether the private key for the certificate resulting from the enrollment will be retained in Keyfactor Command (true) or not (false).	
Management Job Time	The schedule for the management job to add the certificate to any certificate store(s).	

Name	Description	
Metadata	Values for the metadata fields that will be associated with the certificate once it is in Keyfactor Command.	
PFX Password Secret Instance Id	The Keyfactor Command reference ID for the PFX password used to secure the PFX file on download.	
Renewal Certificate	Certificate Id - The Keyfactor Command reference ID of the certificate this certificate replaces on a renewal.	
Renewal Certificate Data: Raw	The certificate that this certificate replaces on a renewal as returned by the CA in base-64 encoded binary format.	
Renewal Certificate Data: Parsed	The certificate details for the certificate that this certificate replaces on a renewal, including: • Issued DN The distinguished name of the certificate. • Issuer DN The distinguished name of the issuer. • Thumbprint The thumbprint of the certificate. • Not After The date, in UTC, on which the certificate expires. • Not Before The date, in UTC, on which the certificate was issued by the certificate authority. • Metadata The metadata fields populated for the certificate.	

SANs: Type	The subject alternate names defined for the certificate. Possible types that can be entered within Keyfactor Command are DNS Name, IPv4 Address, IPv6 Address, User Principal Name, and Email. Within each type is a list of entries for that type shown with a key name of the entry number and the actual value. For example, if you had two DNS SANs, the DNS Name section would look something like:
	`Entry 1: myfirstsan.keyexample.com`
	`Entry 2: mysecondsan.keyexample.com`	

Name	Description	
Stores	The certificate stores to which the certificate should be distributed, if applicable.	
Template	The template that was used when requesting the certificate.	
(Custom)	Optional user-generated custom fields returning response data from PowerShell scripts or REST requests. These will be sorted alphabetically in among the other fields on the page.	

Revocation

For a revocation, this section may include:

Name	Description
Certificate Authority	The certificate authority that issued the certificate.
Certificate Id	The Keyfactor Command reference ID for the certificate being revoked.
Comment	A freeform reason or comment to explain why the certificate is being revoked.
Delegate	A flag indicating whether delegation was enabled for the certificate authority that issued the certificate at the time revocation was requested (true) or not (false). For more information, see Authorization Methods Tab on page 367.
Effective Date	The date and time when the certificate will be revoked.
Initiating User Name	The name of the user who initiated the workflow in DOMAIN\\username format.
Operation Start	The time at which the revocation workflow was initiated.
RevokeCode	The specific reason that the certificate is being revoked. Possible values are: • -1—Remove from Hold • 0—Unspecified • 1—Key Compromised • 2—CA Compromised • 3—Affiliation Changed • 4—Superseded • 5—Cessation of Operation

Name	Description
	• 6—Certificate Hold • 7—Remove from CRL. Only valid in the case that a cert is already on a CRL in a manner that it can be removed, such as Certificate Hold.
Serial Number	The serial number of the certificate being revoked.
Thumbprint	The thumbprint of the certificate being revoked.
(Custom)	Optional user-generated custom fields returning response data from PowerShell scripts or REST requests. These will be sorted alphabetically in among the other fields on the page.

Certificate Entered Collection and Certificate Left Collection

For a certificate that entered or left a certificate collection, this section generally includes:

Name	Description
Certificate Id	The Keyfactor Command reference ID for the certificate added to or removed from the certificate collection.
Initiating User Name	The name of the user who initiated the workflow—generally *Timer Service* in this case.

Signal Input

In the Signal Input section of the page, you can submit one or more signals for the step. For the built-in require approval workflow step type, this is where you send an approval or denial for the request along with a comment about the approval or denial.

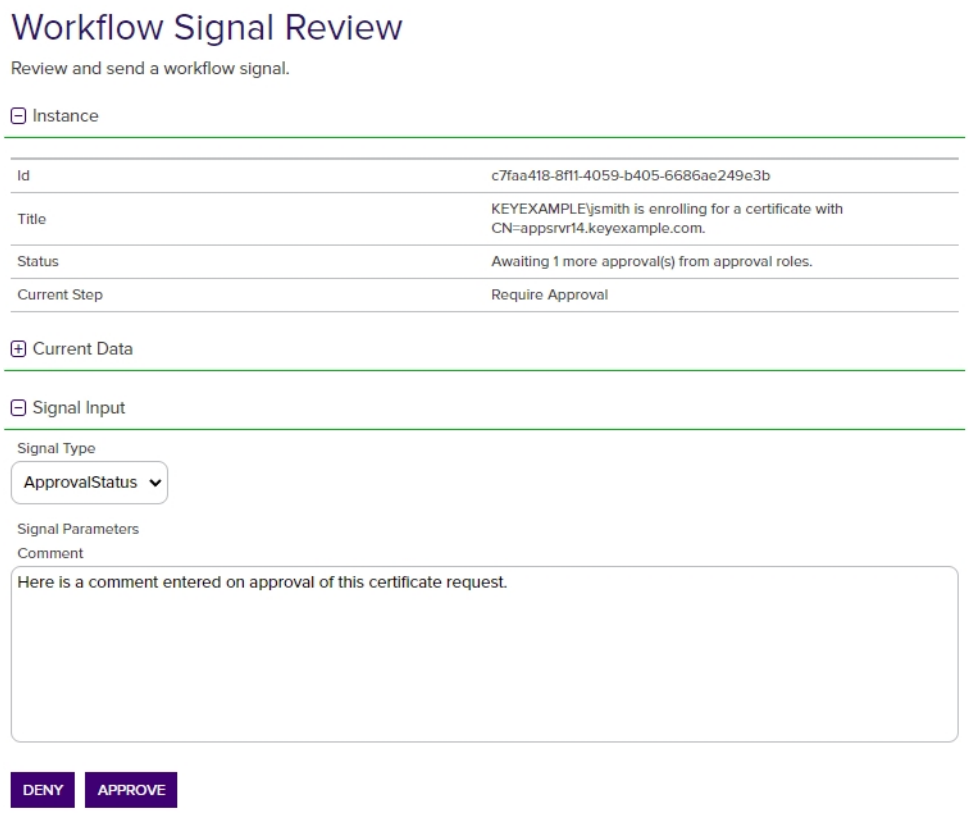


Figure 207: Approve or Deny a Workflow Instance

A custom workflow step requiring signal input may have more than one signal type to select from in the dropdown, may have input fields to submit data with the signal, and will likely have buttons with labels other than *Deny* or *Approve*.

4. At the bottom of the Workflow Signal Review page in the Signal Input section, select an option in the Signal Type dropdown, enter any required signal data, and click an appropriate signal button to submit the signal. For the built-in require approval workflow step type, select *ApprovalStatus* in the dropdown (there is only one choice), enter an optional **Comment** (the maximum comment length is 500 characters), and click either **Approve** to add your approval to the workflow or **Deny** to deny the workflow instance.

> **Tip:** If you reference the approve/deny comments using the $(approvalsignalcmnts) token, the included comments will vary depending on where you use the token. If you use the token in an email message within a require approval step, only comments from that require approval step will be included. If you use the token in a separate email step within the same workflow, all comments from any require approval steps within the workflow will be included.

Important: Comments entered when approving or denying a built-in require approval workflow step can be included in emails delivered either as part of the require approval step or in subsequent steps within the workflow, but they are not retained for future reference. If you would like to retain them for future reference, use a workflow step that copies the comment(s) to a metadata field (see Use Custom PowerShell on page 281).

5. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Note: The workflow definition may require more than one approval to be completed and so may not be immediately completed when you click Approve. However, a single denial is enough to reject the workflow instance.

An audit log entry is created when you provide input to a workflow instance (see Audit Log on page 716).

Using the Workflow Assigned to Me Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

DefinitionId

Complete or partial matches with the Keyfactor Command reference GUID of the workflow *definition*.

Id

Complete or partial matches with the Keyfactor Command reference GUID of the workflow *instance*.

StartDate

The date and time when an instance was initiated.

Status

Status matches or doesn't match the referenced value. Supported statuses are:

- Unknown
- Running
- Suspended
- Failed
- Complete

- Rejected
- CanceledforRestart

InitiatingUserName

Complete or partial matches with the name of the user who initiated the workflow instance in DOMAIN\username format for Active Directory authentication or username for an identity provider other than Active Directory.

LastModified

The date and time on which an initiated instance was last updated. The instance is updated each time a step in the workflow is completed, when signals are received for a step that accepts signals (e.g. a requires approval step), or when an instance is stopped or restarted.

Title

Complete or partial matches with the description for the action taking place in the workflow instance step. The values in the title will vary and generally include the user initiating the request and the CN of the certificate or certificate request involved.

WorkflowType

The type of workflow (enrollment, revocation, certificateenteredcollection, or certificateleftcollection).

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only

support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

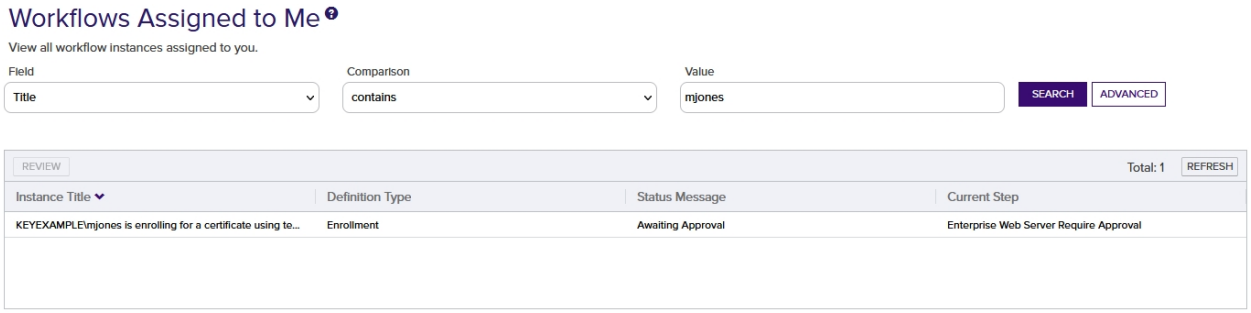


Figure 208: Simple Workflows Assigned to Me Search

The search results can be sorted by clicking on a column header in the results grid. Only the Instance Title column sortable. Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may be adjusted by click-holding and dragging the line separating two column headers.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

Workflows Created by Me Operations

On the Created by Me tab of the My Workflows page you can view all the workflows that the current user initiated.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Workflows > Instances > Read > Mine
Or:
Workflows > Instances > Read

To view details of a workflow instance:

1. In the Management Portal, browse to *Workflow > My Workflows*.
2. On the Created by Me tab of the My Workflows page, double-click or click **View** from either the top or right click menu.
3. On the Workflow Signal Review page, review the information in the instance. Information on the review page includes:

Name	Description
Id	A GUID indicating the Keyfactor Command reference ID for the instance.

Title	A description for the action taking place in the step. For example:
	`"KEYEXAMPLE\jsmith is enrolling for a certificate with CN=appsrvr12.keyexample.com."`
	Or:
	`"KEYEXAMPLE\mjones is revoking certificate with CN=appsrvr14.keyexample.com."`

Status	The current status message of the workflow instance. For example, for an enrollment that **succeeded**, the status message might be: `Issued. The private key was successfully retained.`

Name	Description

	For a workflow **suspended and awaiting approval**, the status message might be:
	`Awaiting 2 more approval(s) from approval roles.`
	For an **enrollment** that could not be submitted because a regular expression rule was not met, the status message might be something like:
	`Pre-process failed: Invalid ST provided: Value must be one of California, Washington, Texas, New York, Illinois or Ohio.`
	For an **enrollment** that failed due to rejection by the CA, the status message might be:
	`The certificate request failed with the reason '[CA reason]'`
	A workflow that failed at a **PowerShell step** might include the PowerShell error in the status message:
	`Step 'Run PowerShell' failed: At line:5 char:19`
	`+ [datetime]$Date`
	`+ ~`
	`Missing ')' in function parameter list.`
	`At line:7 char:1`
	`+ )`
	`+ ~`
	`Unexpected token ')' in expression or statement.`

Current Step	The display name defined for the workflow instance step at which the instance has paused or stopped. For a successfully completed enrollment or revocation workflow, this will be either *Keyfactor-Revoke* or *Keyfactor-Enroll*. For a suspended workflow, this will be the custom step that is awaiting user input to continue the workflow. For a failed workflow, this will be the step at which the workflow failed.

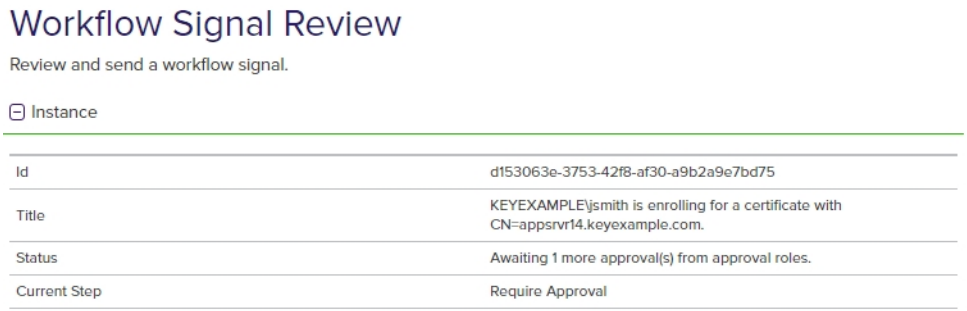


Figure 209: Workflow Instance Review

The data included in this section will vary depending on the request type, the status of the request, and the configuration of the workflow.

Enrollment

Name	Description	
Subject	The distinguished name of the certificate.	
CSR:Raw	The unparsed version of the certificate signing request generated for the certificate request.	
CSR: Parsed	The parsed version of the certificate signing request generated for the certificate request. The CSR may include: • Key Length The desired key size for the certificate. • Key Type The desired key encryption for the certificate. • C The country (two characters) of the certificate. • ST The state or province of the certificate. • L The city or locality of the certificate. • OU The organizational unit of the certificate. • O The organization of the certificate. • E The email address of the certificate.	

Name	Description	
	• CN The common name of the certificate. • DNS Name A SAN value containing a DNS name. • IP Address A SAN value containing an IP v4 or IP v6 address. • RFC822 Name A SAN value containing an email address. • Other name:Principal A SAN value containing a user principal name (UPN).	
Additional Attributes	Values for any custom enrollment fields set on the certificate template to supply custom request attributes to the CA during the enrollment process.	
CA Certificate	The certificate information returned from the CA for the certificate that is being requested, including: • CA Certificate ID The ID assigned to the certificate by the CA. • CA Request ID The ID assigned to the certificate request by the CA. • Status The numeric status for the certificate as returned by the CA. • Certificate Template The certificate template used to issue the certificate. • Revocation Date The revocation date for the certificate as returned by the CA, if applicable (generally not for newly enrolled certificates). • Revocation Reason The revocation reason for the certificate as returned by the CA, if applicable (generally not for newly enrolled certificates). • Archived Key A flag indicating whether the certificate is configured for key archival on the CA (true) or not (false). **Note:** This field is populated only after the certificate has been issued by the CA.	

Name	Description	
CA Certificate Data: Raw	The certificate as returned by the CA in base-64 encoded binary format.	
CA Certificate Data: Parsed	• Issued DN The distinguished name of the certificate. • Issuer DN The distinguished name of the issuer. • Thumbprint The thumbprint of the certificate. • Not After The date, in UTC, on which the certificate expires. • Not Before The date, in UTC, on which the certificate was issued by the certificate authority. • Metadata The metadata fields populated for the certificate.	
CA Certificate Request 1.	The certificate request information returned from the CA for the certificate that is being requested, including: • CA Request ID The ID assigned to the certificate request by the CA. • CSR The certificate signing request for the certificate request as returned by the CA. • Status The status for the certificate as returned by the CA. • Requester Name The requester name on the certificate request as returned by the CA. **Note:** This field is populated only if the certificate request fails at the CA level or requires manager approval at the CA level.	
Certificate Authority	The certificate authority that will be used to enroll against in *host-name\logical name* format.	
Custom Name	A custom friendly name for the certificate, if entered at enrollment.	

Name	Description	
Disposition Message	A message about the certificate request. **Note:** This field is populated only after the certificate request has been submitted to the CA.	
Format	The desired output format for the certificate. A value of STORE indicates that the certificate is intended to be delivered into one or more certificate stores.	
Include Chain	A flag indicating whether to include the certificate chain in the enrollment response (true) or not (false).	
Initiating User Name	The name of the user who initiated the workflow in DOMAIN\username format.	
Is PFX	A flag indicating whether the certificate enrollment type that initiated the workflow instance was PFX (true) or CSR (false).	
Issuer DN	The distinguished name of the issuer.	
Key Retention	A flag indicating whether the private key for the certificate resulting from the enrollment will be retained in Keyfactor Command (true) or not (false).	
Key Status	A numeric value indicating the status of the private key retention for the certificate within Keyfactor Command. Possible values are: 1. • 0—Unknown • 1—Saved • 2—Expected • 3—NoRetention • 4—Failure • 5—Temporary	
Keyfactor Id	The Keyfactor Command reference ID for the certificate.	
Management Job Time	The schedule for the management job to add the certificate to any certificate store(s).	
Metadata	Values for the metadata fields that will be associated with the certificate once it is in Keyfactor Command.	
PFX Password	The Keyfactor Command reference ID for the PFX password used to secure	

Name	Description	
Secret Instance Id	the PFX file on download.	
Private Key Converter	An internally used Keyfactor Command field.	
Renewal Certificate	Certificate Id - The Keyfactor Command reference ID of the certificate this certificate replaces on a renewal.	
Renewal Certificate Data: Raw	The certificate that this certificate replaces on a renewal as returned by the CA in base-64 encoded binary format.	
Renewal Certificate Data: Parsed	The certificate details for the certificate that this certificate replaces on a renewal, including: • Issued DN The distinguished name of the certificate. • Issuer DN The distinguished name of the issuer. • Thumbprint The thumbprint of the certificate. • Not After The date, in UTC, on which the certificate expires. • Not Before The date, in UTC, on which the certificate was issued by the certificate authority. • Metadata The metadata fields populated for the certificate.	

SANs: Type	The subject alternate names defined for the certificate. Possible types that can be entered within Keyfactor Command are DNS Name, IPv4 Address, IPv6 Address, User Principal Name, and Email. Within each type is a list of entries for that type shown with a key name of the entry number and the actual value. For example, if you had two DNS SANs, the DNS Name section would look something like:
	`Entry 1: myfirstsan.keyexample.com`
	`Entry 2: mysecondsan.keyexample.com`	

Name	Description	
Serial Number	The serial number of the certificate.	
Stores	The certificate stores to which the certificate should be distributed, if applicable.	
Template	The template that was used when requesting the certificate.	
Thumbprint	The thumbprint of the certificate.	
(Custom)	Optional user-generated custom fields returning response data from PowerShell scripts or REST requests. These will be sorted alphabetically in among the other fields on the page.	

Revocation

For a revocation, this section may include:

Name	Description
Certificate Authority	The certificate authority that issued the certificate.
Certificate Id	The Keyfactor Command reference ID for the certificate being revoked.
Comment	A freeform reason or comment to explain why the certificate is being revoked.
Delegate	A flag indicating whether delegation was enabled for the certificate authority that issued the certificate at the time revocation was requested (true) or not (false). For more information, see Authorization Methods Tab on page 367.
Effective Date	The date and time when the certificate will be revoked.
Initiating User Name	The name of the user who initiated the workflow in DOMAIN\\username format.
Operation Start	The time at which the revocation workflow was initiated.
RevokeCode	The specific reason that the certificate is being revoked. Possible values are: • -1—Remove from Hold • 0—Unspecified • 1—Key Compromised • 2—CA Compromised

Name	Description
	• 3—Affiliation Changed • 4—Superseded • 5—Cessation of Operation • 6—Certificate Hold • 7—Remove from CRL. Only valid in the case that a cert is already on a CRL in a manner that it can be removed, such as Certificate Hold.
Serial Number	The serial number of the certificate being revoked.
Thumbprint	The thumbprint of the certificate being revoked.
(Custom)	Optional user-generated custom fields returning response data from PowerShell scripts or REST requests. These will be sorted alphabetically in among the other fields on the page.

Certificate Entered Collection and Certificate Left Collection

For a certificate that entered or left a certificate collection, this section generally includes:

Name	Description
Certificate Id	The Keyfactor Command reference ID for the certificate added to or removed from the certificate collection.
Initiating User Name	The name of the user who initiated the workflow—generally *Timer Service* in this case.

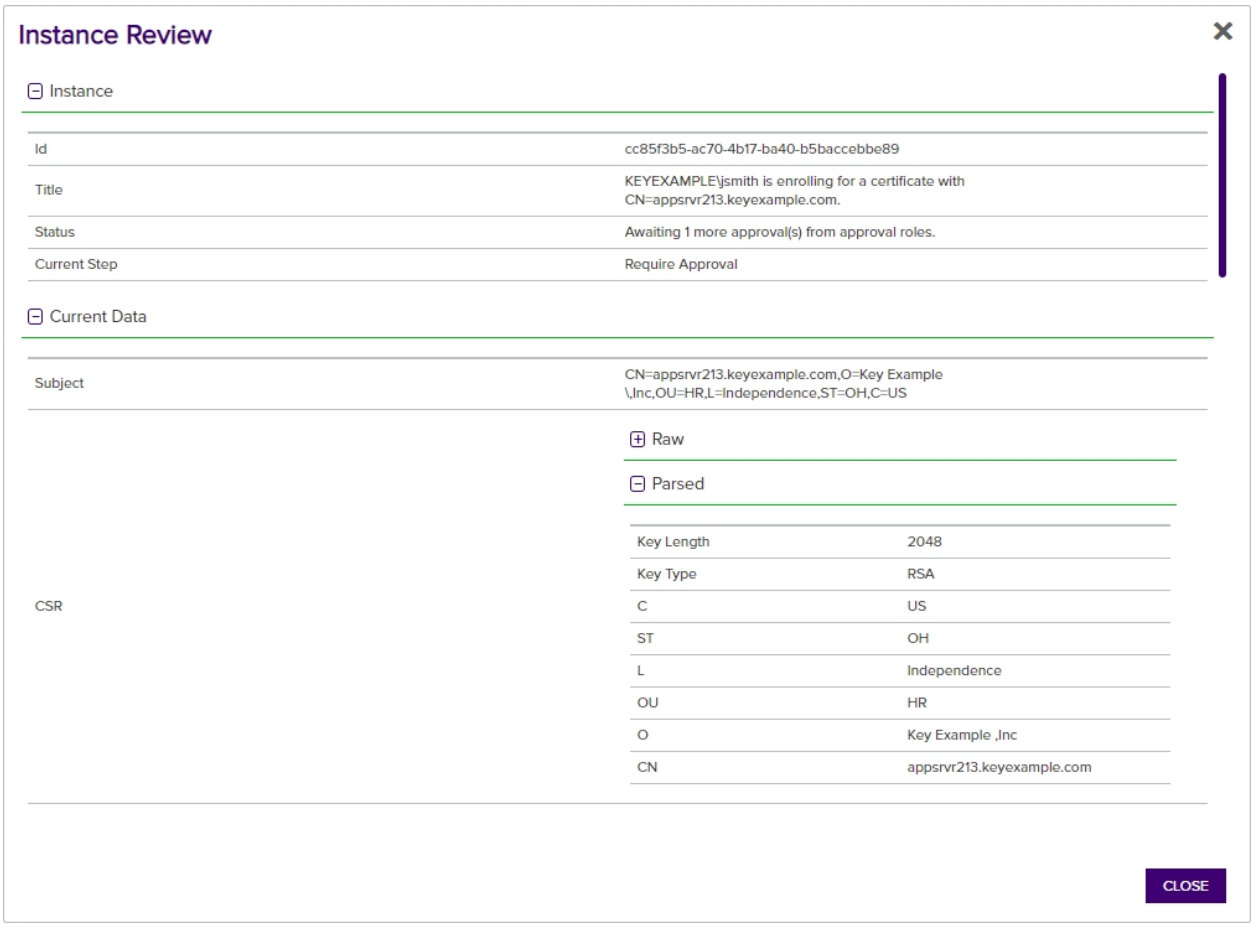


Figure 210: View Details for the Workflow Instance

4. Click **Close** to close the viewer.

Using the Workflow Created by Me Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

DefinitionId

Complete or partial matches with the Keyfactor Command reference GUID of the workflow *definition*.

Id

Complete or partial matches with the Keyfactor Command reference GUID of the workflow *instance*.

InitiatingUserName

Complete or partial matches with the name of the user who initiated the workflow instance in DOMAIN\username format for Active Directory authentication or username for an identity provider other than Active Directory.

LastModified

The date and time on which an initiated instance was last updated. The instance is updated each time a step in the workflow is completed, when signals are received for a step that accepts signals (e.g. a requires approval step), or when an instance is stopped or restarted.

StartDate

The date and time when an instance was initiated.

Status

Status matches or doesn't match the referenced value. Supported statuses are:

- Unknown
- Running
- Suspended
- Failed
- Complete
- Rejected
- CanceledforRestart

Title

Complete or partial matches with the description for the action taking place in the workflow instance step. The values in the title will vary and generally include the user initiating the request and the CN of the certificate or certificate request involved.

WorkflowType

The type of workflow (enrollment, revocation, certificateenteredcollection, or certificateleftcollection).

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

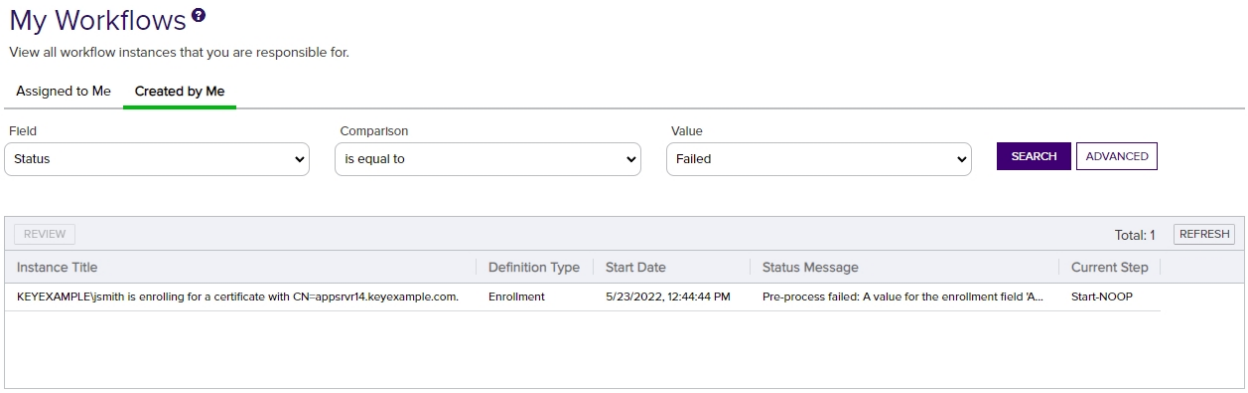


Figure 211: Simple Workflows Created by Me Search

The search results can be sorted by clicking on a column header in the results grid. Only the Instance Title column sortable. Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may be adjusted by click-holding and dragging the line separating two column headers.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

2.1.8 Locations

The options available in the Locations section of the Management Portal are:

- Certificate Authorities on the next page
  Import CAs from Active Directory and/or define CAs, configure synchronization and monitoring tasks for them, set authorization methods and configure enrollment details.
- Certificate Templates on page 379
  Import certificate templates from Active Directory or EJBCA, view certificates, and configure template-specific enrollment details such as: enrollment fields, authorization methods, metadata, template regular expressions, enrollment defaults and policies. Also, set system-wide template enrollment regular expressions, enrollment defaults and policies.
- Certificate Stores on page 408
  Configure paths to certificate stores on multiple machines and devices in the environment, group them into containers for organization, and configure inventory schedules to synchronize the certificates in the stores to Keyfactor Command, and view certificate inventory.
- SSL Discovery on page 453
  Configure SSL endpoint groups on which to run discovery and monitoring jobs and then import certificates from the endpoints for monitoring, reporting and alerting purposes. Define orchestrator pools and view scan results.

2.1.8.1 Certificate Authorities

Your Microsoft and EJBCA certificate authorities (CAs) are defined in the Management Portal to support synchronization to the Keyfactor Command database and support enrollment. Microsoft CAs in the local forest in which Keyfactor Command is installed or in a forest in a two-way trust with this forest may be imported from Active Directory or manually configured. Other Microsoft CAs and EJBCA CAs need to be manually configured. During initial provisioning, any domain-joined Microsoft CAs in the primary Active Directory forest will be imported automatically by the Keyfactor Command configuration wizard.

Important: In order for CAs to successfully synchronize to the Keyfactor Command database and perform other functions (e.g. enrollment), the service account under which Keyfactor Command is making the request to the CA must be granted appropriate permission to the CA database as per Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832.

CAs that need to be added manually include:

- A domain-joined enterprise or standalone Microsoft CA in a forest with a one-way trust (either direction) with the forest in which Keyfactor Command is installed
- A domain-joined enterprise or standalone Microsoft CA in a forest that has no trust with the forest in which Keyfactor Command is installed
- An EJBCA CA
- A non-domain-joined standalone Microsoft CA
- A Keyfactor CA gateway in the forest in which Keyfactor Command is installed that has not been registered in Active Directory

  The CA gateways are used to access cloud certificate providers (e.g. Entrust) or to support Microsoft or EJBCA CAs in remote or cloud environments (e.g. the Cross-Forest Gateway).

  Note: Keyfactor CA gateways are not supported in any configuration other than in the same forest in which Keyfactor Command is installed.

- An on-premise Microsoft CA accessed via the Keyfactor CA Management Gateway using a managed instance of Keyfactor Command
- An on-premise EJBCA CA accessed via the Keyfactor CA Management Gateway using a managed instance of Keyfactor Command
- A Microsoft CA accessed via the Keyfactor Universal Orchestrator

  Note: You must install and configure the Keyfactor Universal Orchestrator on a machine in the same forest where the Microsoft CA resides and configure it with CA Support and approve the orchestrator in the Management Portal before creating the CA record.

Note: All CAs need to be added manually if you're using Keyfactor Command on a non-domain-joined server.

CAs that need to be configured manually include:

- Domain-joined enterprise or standalone Microsoft CA in a forest with a one-way trust (either direction) with the forest in which Keyfactor Command is installed
- Domain-joined enterprise or standalone Microsoft CA in a forest that has no trust with the forest in which Keyfactor Command is installed
- EJBCA CA
- Non-domain-joined standalone Microsoft CA
- Keyfactor CA gateway in the forest in which Keyfactor Command is installed

  The CA gateways are used to access cloud certificate providers (e.g. the Entrust CA Gateway) or to support Microsoft CAs in remote or cloud environments (e.g. the Cross-Forest Gateway).

  Note: Keyfactor CA gateways are not supported in any configuration other than in the same forest in which Keyfactor Command is installed.

- On-premise Microsoft CA accessed via the Keyfactor CA Management Gateway using a managed instance of Keyfactor Command
- On-premise EJBCA CA accessed via the Keyfactor CA Management Gateway using a managed instance of Keyfactor Command
- Microsoft CA accessed via the Keyfactor Universal Orchestrator

  Note: You must install and configure the Keyfactor Universal Orchestrator on a machine in the same forest where the Microsoft CA resides and configure it with CA Support and approve the orchestrator in the Management Portal before creating the CA record.

The majority of CA-related functions within Keyfactor Command are supported by both EJBCA and Microsoft CAs. Table 12: CA Function Matrix includes a list of CA-related functions and the support provided by EJBCA and Microsoft CAs.

Important: EJBCA integration with Keyfactor Command requires EJBCA version 7.8.1 or higher.

Table 12: CA Function Matrix

	EJBCA CA	Microsoft CA
CA Synchronization	✓	✓

	EJBCA CA	Microsoft CA
Template[1] Import	✓	✓
CA Threshold Monitoring (Issuance)	✓	✓
CA Threshold Monitoring (Failures)		✓
CA Health Monitoring	✓	✓
Certificate Enrollment (PFX)	✓	✓
Certificate Enrollment (CSR)	✓	✓
Certificate Revocation	✓	✓
CRL Publishing Following Certificate Revocation	✓	✓
Keyfactor Command Private Key Retention and Key Recovery	✓	✓
CA-Level Key Archiving (* no longer supported as of Keyfactor Command v10)		
CA-Level Key Recovery		✓
Approvals in Workflow Builder	✓	✓
CA-Level Approvals with Pending, Issued and Denied Alerts		✓
Supports use of *Restrict Allowed Requesters* for access control	✓	✓
Requires use of *Restrict Allowed Requesters* for access control	✓	
Requests to the CA can be done in the context of the user initiating the request		✓

[1]When EJBCA templates are imported, they are named using a naming scheme of <end entity profile name>_<certificate profile name> for the template name (short name). New templates do not need to be created for Keyfactor Command.

	EJBCA CA	Microsoft CA
Requests to the CA can be done in the context of a single service account[1]	✓	✓
Supports use of Universal Orchestrator to access remote CA		✓

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Certificate Authority Operations

During installation of Keyfactor Command, CA records are created for any Microsoft CAs found in the local forest in which Keyfactor Command is installed. If you have Microsoft CAs in separate forests in a two-way trust with the forest in which Keyfactor Command is installed, you will need to use the import option to import CA records from those forests. If you have Microsoft CAs in any other configuration or EJBCA CAs, you will need to manually configure CA records for them.

Importing Trusted Forest CAs

Microsoft CA and Keyfactor CA gateway records from the Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest may be imported using the Import option provided they are registered in Active Directory.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Agents > Management > Read
Certificate Authorities > Read
Certificate Authorities > Modify

To import CA records:

1. In the Management Portal, browse to *Locations* > *Certificate Authorities*.
2. On the Certificate Authorities grid, click the **Import** action button to import local or two-way trusted forest CAs and Keyfactor CA gateways.

[1]For EJBCA, this is the end entity associated with the client certificate used to authenticate to the EJBCA CA.

3. In the Import Certificate Authorities dialog, select the forest from which you want to import in the dropdown and click **Import**.

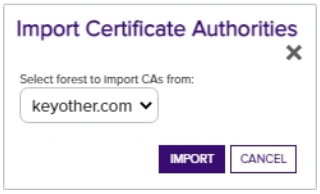


Figure 212: Import Certificate Authorities

Your certificate authorities and CA gateways will be retrieved from Active Directory in the selected forest and will populate the CA grid. Import once for each forest containing Microsoft CAs that you want to synchronize or use for enrollment.

Once the records are imported, use the **Edit** option (see Adding or Modifying a CA Record on the next page) to configure synchronization and other optional settings for the CA.

Tip: This step does not need to be completed for the forest in which Keyfactor Command is installed because those records are imported during the installation process for domain-joined servers unless you have added a new CA or CA gateway following Keyfactor Command installation.

Note: Keyfactor CA gateways are not supported in any configuration other than in the same forest in which Keyfactor Command is installed.

Note: The import option only works for Microsoft CAs or Keyfactor CA gateways that have been registered in Active Directory. Non-domain-joined servers will not have any options available in this dropdown.

Testing a CA Connection

CA connections can be tested using the action buttons on the Certificate Authority dialog, **Test Connection** and **Save and Test**.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Agents > Management > Read
Certificate Authorities > Read
Certificate Authorities > Modify

Certificate Authorities will be tested before they are saved to the database. If the CA can't be verified, an error message with an explanation of the issue will be displayed and added to the Audit Log and you will be given the option to save the configuration anyway.

- For EJBCA, the test checks that the CA name provided is valid for the given EJBCA instance. It validates the hostname, enabled APIs, and authentication certificate. The version is validated (7.8.1 or greater) and connecting to both the REST v1 and SOAP APIs is also validated.
- For Microsoft, the test checks the forest, logical name, CA host, and explicit credentials by using a certutil ping.
- Remote CAs (managed by an orchestrator) will not have the connection tested before saving the CA. The **Test Connection** button will be active, but you will receive a message that the connection cannot be tested if you click it. The **Save and Test** button will skip the test when saving.

To test a CA record:

1. In the Management Portal, browse to *Locations > Certificate Authorities*.
2. On the Certificate Authorities grid, click **Add** to add a new CA, or click **Edit** to modify an existing CA, from either the top or right-click menu.
3. Follow the instructions for adding or modifying a CA (see Adding or Modifying a CA Record below). Once you have entered the details you want to test, click **Test Connection** or **Save and Test**.

    a. Upon a successful test, you will receive a green success notification at the bottom of the page and will be returned to the certificates authority grid.

    b. Upon a test failure, you will receive a pop-up message and an entry will also be added to the audit log. Click **OK** to save the CA despite the error, or click **CANCEL** to return to the dialog to correct any issues.

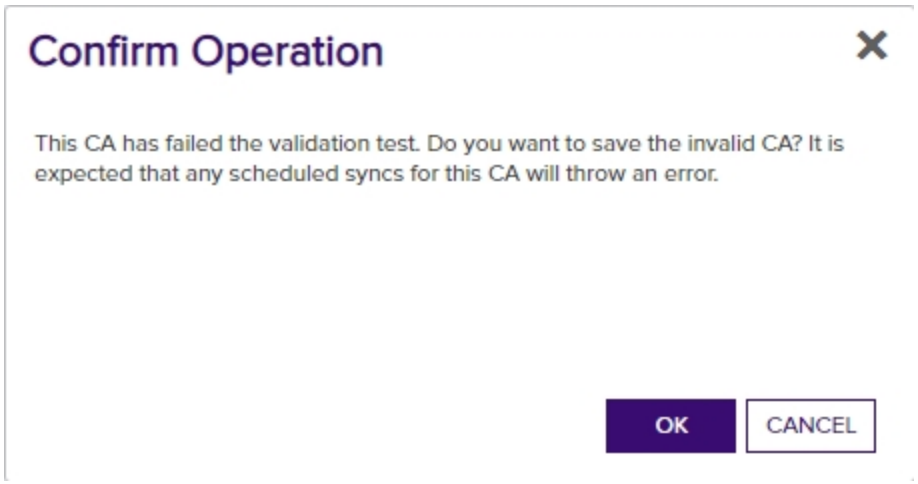


Figure 213: Save Certificate Authority

Adding or Modifying a CA Record

Whether your CA has been imported or added manually, you'll need to update it to configure synchronization and other optional settings. When adding or editing your CAs, the connection can now be tested prior to saving.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Agents > Management > Read
Certificate Authorities > Read
Certificate Authorities > Modify

CAs that need to be added manually include:

- A domain-joined enterprise or standalone Microsoft CA in a forest with a one-way trust (either direction) with the forest in which Keyfactor Command is installed
- A domain-joined enterprise or standalone Microsoft CA in a forest that has no trust with the forest in which Keyfactor Command is installed
- An EJBCA CA
- A non-domain-joined standalone Microsoft CA
- A Keyfactor CA gateway in the forest in which Keyfactor Command is installed that has not been registered in Active Directory

  The CA gateways are used to access cloud certificate providers (e.g. Entrust) or to support Microsoft or EJBCA CAs in remote or cloud environments (e.g. the Cross-Forest Gateway).

  Note: Keyfactor CA gateways are not supported in any configuration other than in the same forest in which Keyfactor Command is installed.

- An on-premise Microsoft CA accessed via the Keyfactor CA Management Gateway using a managed instance of Keyfactor Command
- An on-premise EJBCA CA accessed via the Keyfactor CA Management Gateway using a managed instance of Keyfactor Command
- A Microsoft CA accessed via the Keyfactor Universal Orchestrator

  Note: You must install and configure the Keyfactor Universal Orchestrator on a machine in the same forest where the Microsoft CA resides and configure it with CA Support and approve the orchestrator in the Management Portal before creating the CA record.

Note: All CAs need to be added manually if you're using Keyfactor Command on a non-domain-joined server.

If your Microsoft CA or Keyfactor CA gateway is domain-joined in the forest in which Keyfactor Command is installed or a forest in a two-way trust with this forest and has been registered in Active Directory, you can opt to add a record for it manually, but it is generally easier to use the import option (see Importing Trusted Forest CAs on page 352).

Important: In order for CAs to successfully synchronize to the Keyfactor Command database and perform other functions (e.g. enrollment), the service account under which Keyfactor Command is making the request to the CA must be granted appropriate permission to the CA database as per Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832.

To create a CA record manually or edit an existing one:

1. In the Management Portal, browse to *Locations* > *Certificate Authorities*.
2. On the Certificate Authorities grid, click **Add** to add a new CA, or click **Edit** from either the top or right-click menu to modify an existing one.
3. At the top of the dialog, choose an appropriate CA communication protocol in the **Select CA Communication Protocol** dropdown. The options are:

   - DCOM—Select this option for Microsoft CAs and CA gateways.
   - HTTPS—Select this option for EJBCA CAs and AnyCAGateway REST.

   This field cannot be modified on an edit.
4. The remainder of the Certificate Authority dialog shows four tabs. Only the first three are used for EJBCA CAs. Complete the Certificate Authority dialog with the appropriate data using the following instructions:

The Basic Tab

Details

In the *Details* section populate the **Logical Name**, **Host Name** and **Configuration Tenant** fields with the appropriate information for the CA. (The **Enforce Unique DN** and **Create new End Entity when renewing or reissuing certificates** checkboxes apply only to the HTTPS Certificate Authorities).

The **Configuration Tenant** field cannot be modified on an edit.

Tip: Previous versions of Keyfactor Command referred to the **Configuration Tenant** as the **Template Forest**.

Domain-Joined Enterprise or Standalone Microsoft CA in a Forest with a One-Way Trust (either direction) with the Forest in which Keyfactor Command is Installed

- **Logical Name**—The logical name of the CA in the remote forest. For example: Corp2IssuingCA1

- **Host Name**—The fully qualified domain name of the server on which the CA in the remote forest is installed. For example: corp2ca01.keyother.com
- **Configuration Tenant**—The DNS domain name for the Active Directory forest in which the CA resides. For example: keyother.com

Domain-Joined Enterprise or Standalone Microsoft CA in a Forest that has No Trust with the Forest in which Keyfactor Command is Installed

- **Logical Name**—The logical name of the CA in the remote forest. For example: Corp3IssuingCA1
- **Host Name**—The fully qualified domain name of the server on which the CA in the remote forest is installed. For example: corp3ca01.keyother2.com
- **Configuration Tenant**—The DNS domain name for the Active Directory forest in which the CA resides. For example: keyother2.com

EJBCA CA

- **Logical Name**—The logical name of the EJBCA CA. For example: CorpCA1

  Note: EJBCA CA logical names are case sensitive (e.g. CorpCA1 is not the same as CORPCA1).

- **Host URL**—The URL pointing to the EJBCA CA. For example: https://ejbca01.keyother3.com. If the URL provided does not have a virtual directory (/ejbca or otherwise) the /ejbca will be provided, otherwise it will use what is supplied in the URL.
- **Configuration Tenant**—A reference ID for the EJBCA CA server. For EJBCA CAs, this does not need to be the DNS domain name. The short hostname of the EJBCA CA server makes a good reference ID.

  Important: EJBCA and Microsoft CAs cannot be configured with the same *Configuration Tenant*, so do not set this to the DNS domain name if you will also be configuring Microsoft CAs in the same DNS domain.

- **Enforce Unique DN**
  Checking this will cause Keyfactor Command, upon enrollment, to search the EJBCA CA for end entities with DNs that match the DN in the certificate request. If a matching DN is found, the process will update the existing end entity in EJBCA with the new certificate request information rather than creating a new end entity. If you enable this option in Keyfactor Command, it must also be enabled on the matching EJBCA CA. A mismatch in these settings can result in enrollment failures.

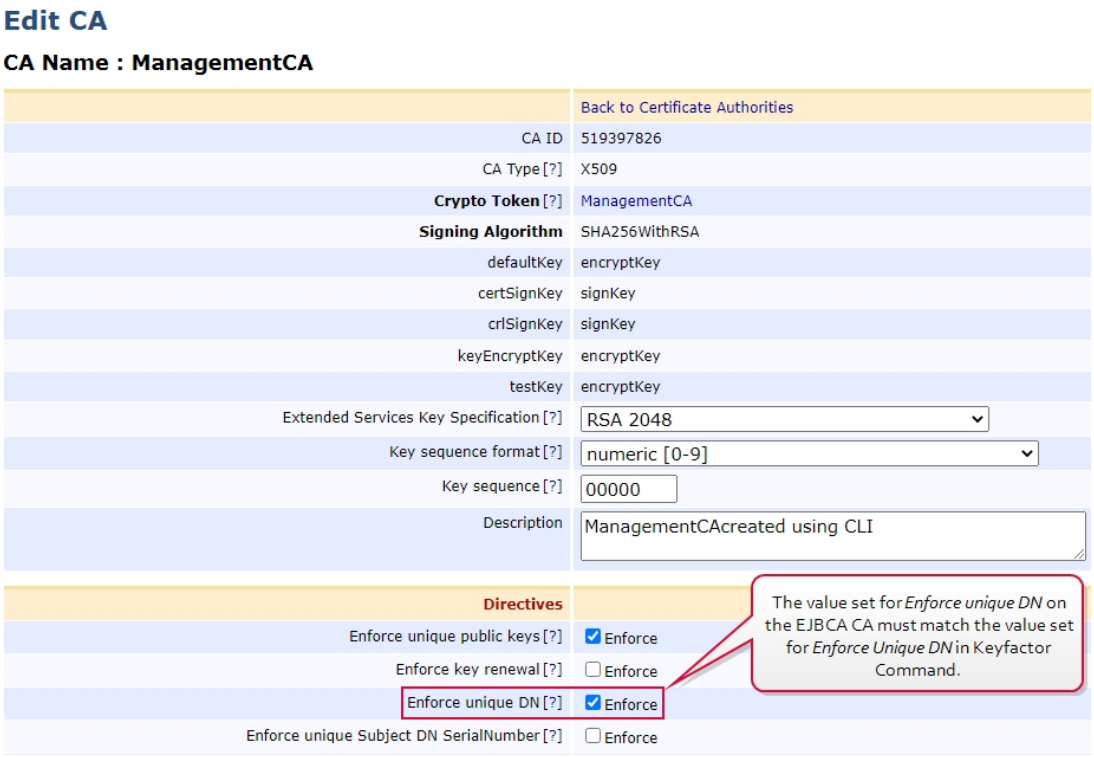


Figure 214: Enforce unique DN Setting on the EJBCA CA

The value of the Keyfactor Command **Enforce Unique DN** setting is verified for each certificate request:

- If unset, enrollment proceeds as usual.
- If set, EJBCA is searched for an end entity associated with the DN and CA in the certificate request and:
  - If none is found, the enrollment proceeds as usual.
  - If one or more is found, the end entity in EJBCA is updated with the information from the certificate request, so that the new certificate request is tied to the same end entity as the existing certificate (or the first one found, if multiple are found). A new password is generated and the enrollment proceeds as usual.

Non-Domain-Joined Standalone Microsoft CA

- **Logical Name**—The logical name of the standalone CA. For example: CorpSARootCA1
- **Host Name**—The fully qualified domain name of the server on which the standalone CA is installed. For example: saroot01.keyexample.com

- **Configuration Tenant**—The DNS domain name for the standalone CA. For example: keyexample.com

Remote CA Accessed via a Keyfactor Universal Orchestrator

- **Logical Name**—The logical name of the CA in the remote forest to which the orchestrator will be connecting for synchronization. For example: Corp4IssuingCA1
- **Host Name**—The fully qualified domain name of the CA in the remote forest to which the orchestrator will be connecting for synchronization. For example: corp4ca01.keyother4.com
- **Configuration Tenant**—The DNS domain name for the Active Directory forest in which the orchestrator is operating and in which the CA resides. For example: keyother4.com

Note: You must install and configure the Keyfactor Universal Orchestrator on a machine in the same forest where the CA resides, configure it with CA Support and approve the orchestrator in the Management Portal before creating the CA record.

Domain-Joined Enterprise or Standalone Microsoft CA in the Forest in which Keyfactor Command is Installed

- **Logical Name**—The logical name of the CA in the local forest. For example: CorpIssuingCA1
- **Host Name**—The fully qualified domain name of the server on which the CA in the local forest is installed. For example: corpca01.keyexample.com
- **Configuration Tenant**—The DNS domain name for the Active Directory forest in which the CA resides. For example: keyexample.com

Keyfactor CA Gateway

- **Logical Name**—The logical name of the CA gateway in the local forest. For example: EntrustGateway
- **Host Name**—The fully qualified domain name of the server on which the CA gateway in the local forest is installed. For example: entgtw1.keyexample.com
- **Configuration Tenant**—The DNS domain name for the Active Directory forest in which the CA resides. For example: keyexample.com

Keyfactor CA Management Gateway

- **Logical Name**—The logical name created when the gateway was configured. The logical name is unique for each CA gateway. For a gateway providing a bridge to an on-premise Microsoft CA, the name configured as the gateway logical name should match the logical name of the Microsoft CA.

- **Host Name**—The fully qualified domain name of the server in the managed forest environment in which the Keyfactor CA Management Gateway is installed.
- **Configuration Tenant**—The DNS domain for the Active Directory forest in the managed forest environment in which the Keyfactor CA Management Gateway is installed.

AnyCAGateway REST

- **Logical Name**—The logical name, exactly as entered in AnyCA Gateway portal. The logical name is unique for each CA.

  Note: Logical names are case sensitive (e.g. CorpCA1 is not the same as CORPCA1).

- **Host URL**—The fully qualified domain name of the server on which the gateway is installed and the port number defined in -ServerPort. For example: CAGateway24.keyexample.com:8443. The Host URL is unique for each CA.
- **Configuration Tenant**—A reference ID for the gateway CA. Can be any name you choose.
- **Create new End Entity when renewing or reissuing certificates**

  Checking this will force the gateway CA to create a new end entity for any renew/reissue of a certificate. This is **required** for the AnyCAGateway REST.

Scan

In the *Scan* section, choose when to schedule full and incremental scans. You can choose to run each scan **Weekly**, **Daily** or on an **Interval**:

- If you select **Weekly**, you can select one or more days of the week on which to run the scan and a time when the scan should begin.
- If you select **Daily**, you can set the time of day when the scan should begin.
- If you select **Interval**, you can select a scan frequency of anywhere from every 1 minute to every 12 hours.
- Select **Off** in the dropdown to disable a scan job.

  There are two types of synchronization schedules available for CAs—Full and Incremental. You do not necessarily need to configure both types. A full scan reads all the certificates and certificate requests in the CA database and synchronizes them to Keyfactor Command regardless of their current state in Keyfactor Command. An incremental scan reads the certificates and certificate requests in the CA database that have been generated since the last full or incremental scan and synchronizes them to Keyfactor Command.

A common configuration would be a full scan once or twice a week to provide a clean image of the CA database with a frequent incremental scan to provide timely updates to Keyfactor Command. For a large CA database, a full scan can take a long time to complete. Since an incremental scan only synchronizes updates that have occurred to the CA database since the last synchronization was run, this process is generally quick (other than for the initial synchronization when Keyfactor Command is first installed). The frequency of the incremental scans would depend on the volume of certificate requests coming into the CA.

Note: For EJBCA CAs, if the certificate profile has a *Validity Offset* configured to a value greater than the value configured in the *CA Sync Backward Offset Minutes* application setting (15 minutes by default), certificates requested outside of Keyfactor Command will not be picked up on incremental scans. These certificates will only appear in Keyfactor Command on a full synchronization. The *CA Sync Backward Offset Minutes* application setting should be set to the same number of minutes as the *Validity Offset* value, if *Validity Offset* is configured.

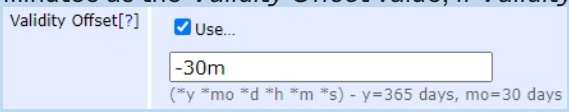


Figure 215: EJBCA Certificate Profile Validity Offset Greater than 15 Minutes

Note: For EJBCA CAs, if the certificate profile has *Allow Backdated Revocation* configured and a revocation is completed outside of Keyfactor Command with a backdate of greater than 10 minutes, the revocation will not be picked up on incremental scans. These revocations will only appear in Keyfactor Command on a full synchronization.

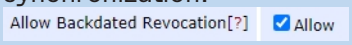


Figure 216: EJBCA Certificate Profile Backdated Revocation

For Microsoft CAs, if desired check the **Sync External Certificates** box to allow foreign certificates that have been imported into a Microsoft CA to be synchronized to Keyfactor Command along with the certificates issued by the Microsoft CA. This option does not appear for HTTPS CAs.

Enrollment

In the *Enrollment* section, check the **Enable PFX Enrollment** and/or **Enable CSR Enrollment** box to enable enrollment for the CA through Keyfactor Command.

If you wish to use *One-Click Renewal* for certificates, the **Allow One-Click Renewals** option must be enabled in both the templates and CAs to which you want *One-Click Renewal* to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354). For more information about one-click renewals, see Renew on page 69.

Note: In order to perform enrollment through Keyfactor Command, the account making the request to the CA must be granted appropriate enroll permissions on the CA itself. Which account this is depends on the authorization configuration (see Authorization Methods Tab on page 367):

- If **Use Explicit Credentials** is set to *true* (box checked), enrollment is done in the context of that explicit user and that user needs permission.
- If **Use Explicit Credentials** is set to *false* (box not checked), the user context in which enrollment is done will vary (see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832).

Enrollment is not supported using NTLM authentication.

If desired, check the **Require Subscriber Terms** box to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling. Configure a link to the custom terms using the *URL to Subscriber Terms* application setting (see Application Settings: Enrollment Tab on page 609).

Tip: To fully configure enrollment for the CA, you will also need to configure access on the Authorization Methods tab (see Authorization Methods Tab on page 367) and configure templates (see Certificate Template Operations on page 381).

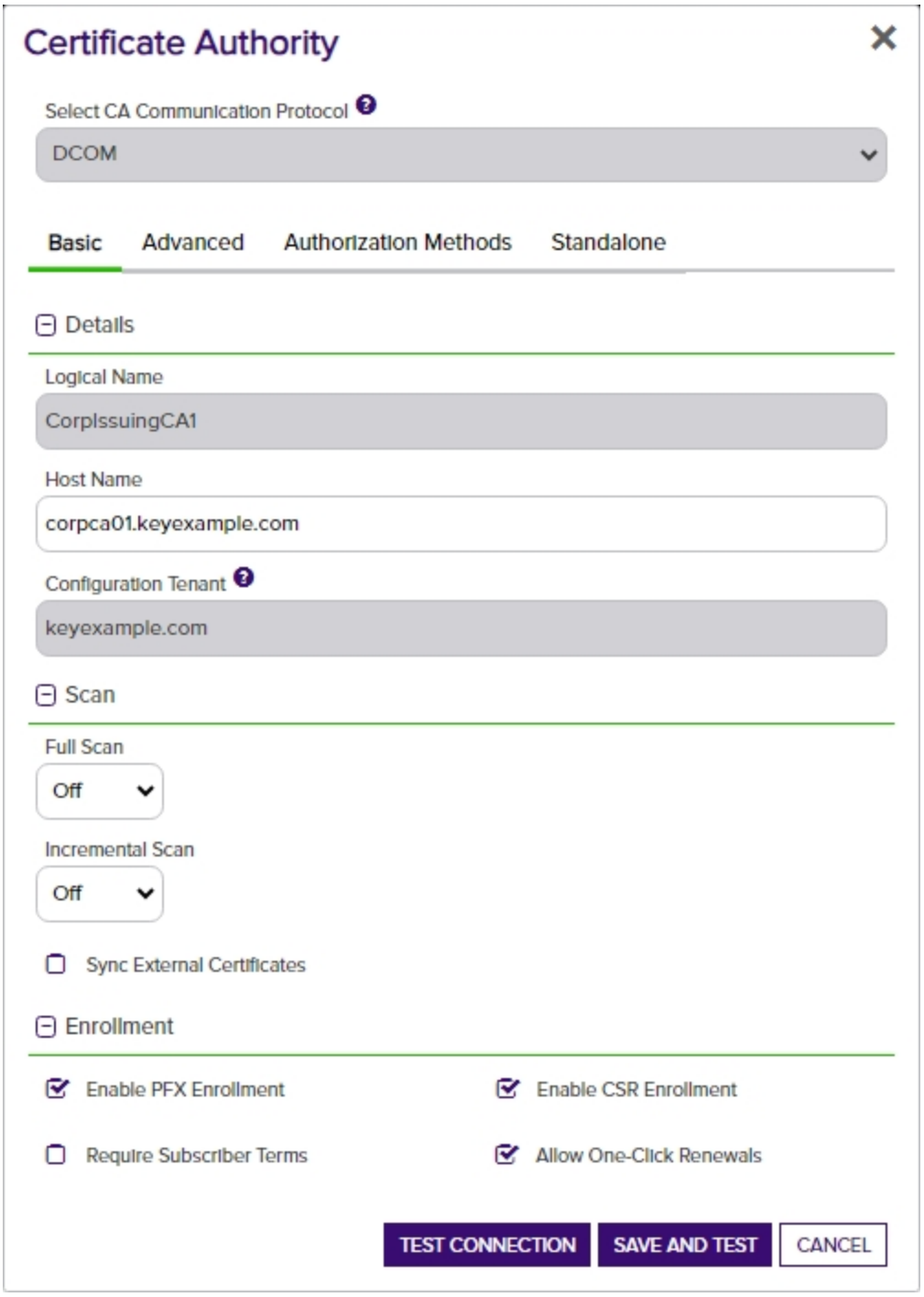


Figure 217: Certificate Authority Basic Tab for a Microsoft CA

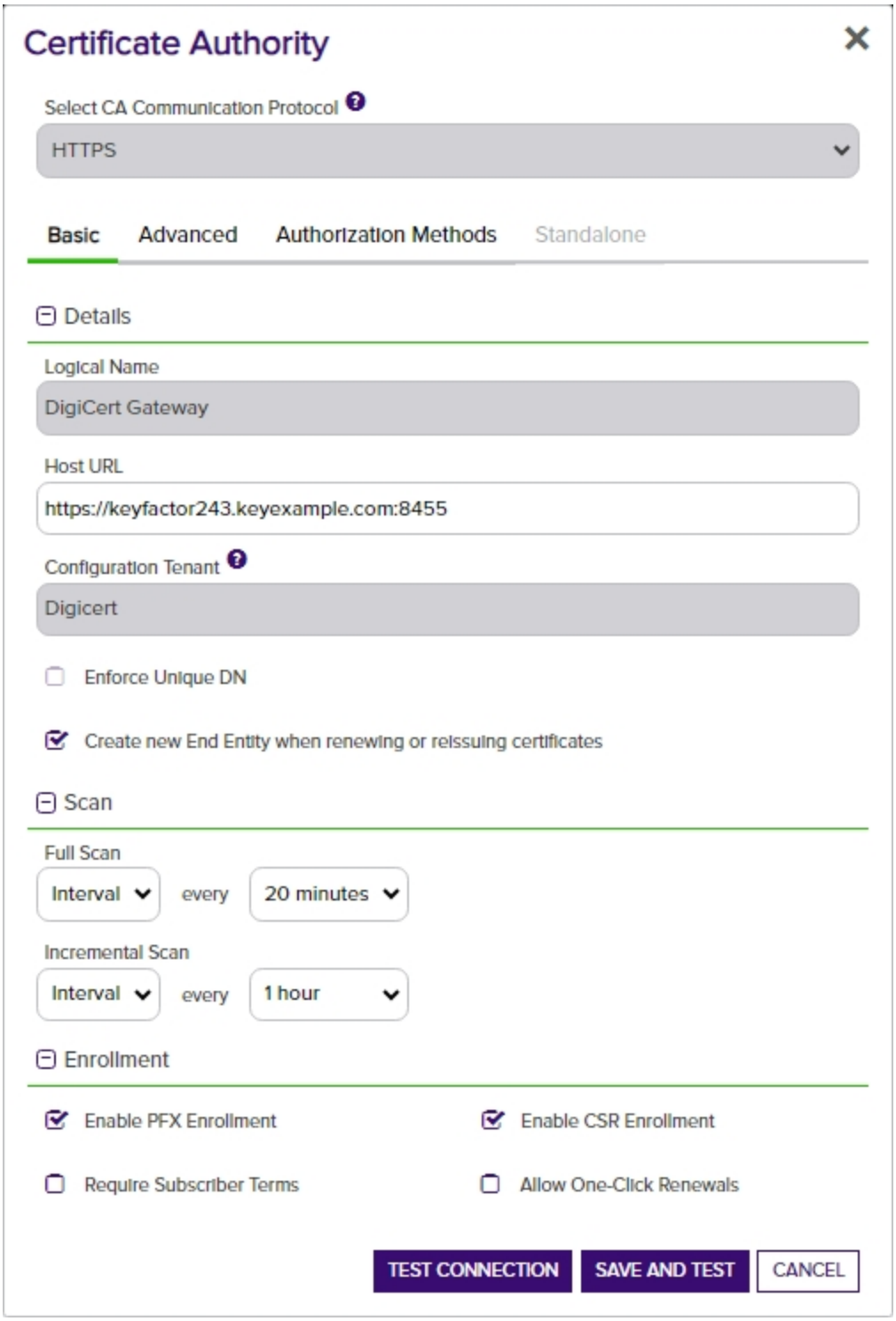

Certificate Authority
Select CA Communication Protocol
HTTPS
Basic
Advanced
Authorization Methods
Standalone
Details
Logical Name
DigiCert Gateway
Host URL
https://keyfactor243.keyexample.com:8455
Configuration Tenant
Digicert
Enforce Unique DN
Create new End Entity when renewing or reissuing certificates
Scan
Full Scan
Interval
every
20 minutes
Incremental Scan
Interval
every
1 hour
Enrollment
Enable PFX Enrollment
Enable CSR Enrollment
Require Subscriber Terms
Allow One-Click Renewals
TEST CONNECTION
SAVE AND TEST
CANCEL

Figure 218: Certificate Authority Basic Tab for an AnyGateway REST CA

Note: When configuring an HTTPS CA, **Enforce Unique DN** and **Create new End Entity when renewing or reissuing certificates** are mutually exclusive options; they CANNOT both be selected since they're telling the CA to do opposite things.

- **Enforce Unique DN** will be DISABLED if **Create new End Entity when renewing or reissuing certificates** is ENABLED.
- **Create new End Entity when renewing or reissuing certificates** will be DISABLED if **Enforce Unique DN** is ENABLED.
- **Create new End Entity when renewing or reissuing certificates** is set to false by default.

Advanced Tab

Details

In the *Details* section, if you've opted to use the Keyfactor Universal Orchestrator to communicate with a remote CA, check the **Use Orchestrator** box and choose the appropriate orchestrator from the dropdown.

Note: The Orchestrator dropdown is only active if the **Use Orchestrator** box is checked. If **Use Orchestrator** is checked, the Orchestrator dropdown will populate with any orchestrators approved in Keyfactor Command with the CA capability. The Keyfactor Universal Orchestrator must be installed on a machine in the forest where the remote CA resides, installed and configured as per Universal Orchestrator on page 2879. In addition, in the Management Portal, the Keyfactor Universal Orchestrator must be configured as per Orchestrator Management on page 496.

Monitoring

In the *Monitoring* section, check the **Enable Monitoring** box to turn on email alerting when certificate issuance or failures (including denials) since the last threshold alert was sent falls outside the configured limits. You can choose to schedule the alerts either for daily delivery at a specified time or at intervals of anywhere from every 1 minute to every 12 hours. Daily is the most common configuration. Set the thresholds for:

- **Issuance Greater Than**—You will receive an alert if more certificates are issued by this CA in the time period between executions of the alert than the number you set here. The value set here must be greater than, or equal to, the value set for *Issuance Less Than*.

- **Issuance Less Than**—You will receive an alert if fewer certificates are issued by this CA in the time period between executions of the alert than the number you set here. The minimum allowed value for *Issuance Less Than* is 1.
- **Failures Greater Than**—You will receive an alert if more certificate requests fail or are denied by this CA in the time period between executions of the alert than the number you set here. Zero is a valid setting (meaning you will receive an alert for a single failure).

  Note: EJBCA CAs do not return failure counts using the API, so failures cannot be reported with threshold monitoring for EJBCA CAs.

In addition to configuring the thresholds for each CA, you must also configure the email recipients on the Alert Recipients tab (see Certificate Authority Monitoring on page 378) of the Certificate Authorities page. Monitoring is not supported for CAs accessed with the Keyfactor Universal Orchestrator.

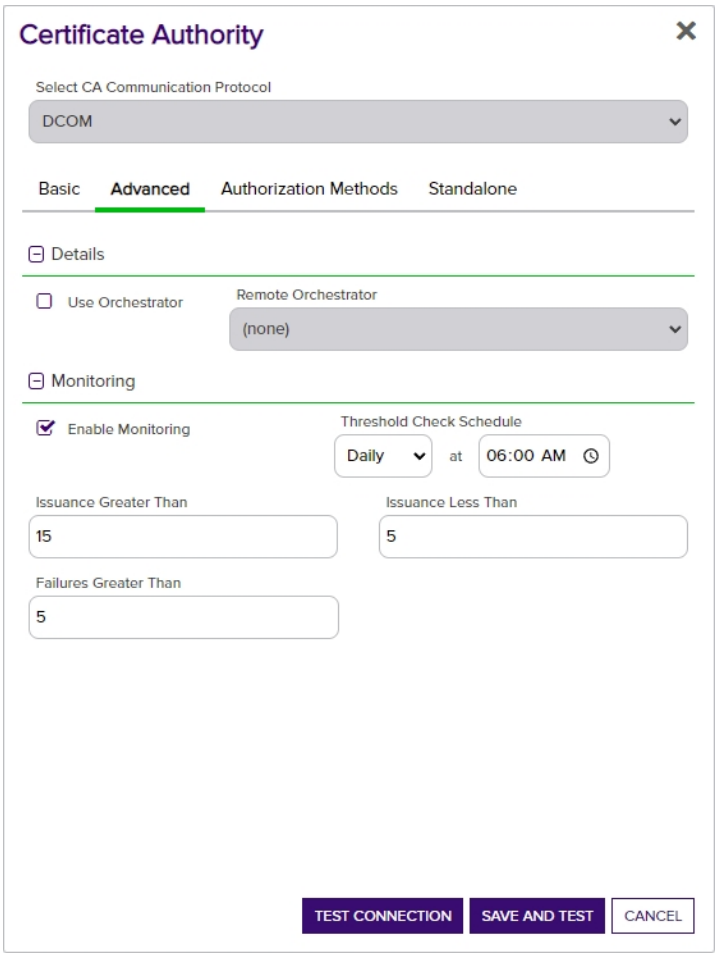


Figure 219: Certificate Authority Advanced Tab for Microsoft CA

Authorization Methods Tab

On the Authorization Methods tab, you configure how access for management tasks and enrollment occurs for the CA.

Tip: Keyfactor recommends the following configuration for most DCOM CAs to support access control within Keyfactor Command:

- **Use Explicit Credentials**: True or false as required by the environment
- **Delegate Management Operations**: False (box unchecked)
- **Delegate Enrollment**: False (box unchecked)
- **Restrict Allowed Requesters**: Set to the Keyfactor security roles allowed to perform certificate enrollment for this CA. If you're using workflow (see Workflow

Definitions on page 230), the users who hold these roles are the ones who are able to initiate workflows. This is entirely separate from the roles configured within workflows, which control the users who are able to approve workflows.

Note: If **Use Explicit Credentials**, **Delegate Management Operations** and **Delegate Enrollment** are all set to *false* (box unchecked), requests to the CA are made in the context of the Keyfactor API application pool user. For more information, see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832.

Use Explicit Credentials (DCOM-Microsoft CAs)

The **Use Explicit Credentials** option allows you to configure specific credentials that will be used to make requests to the CA for management tasks and enrollment. This is generally used for Microsoft CAs where Windows integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest.

To configure this option, check the **Use Explicit Credentials** box and enter a username in the format DOMAIN\username for a service account user in the forest in which the CA resides or, for non-domain-joined machines, a local machine account on the machine on which the CA is installed. Click the **Set Explicit Password** button and in the Set Explicit Password dialog, choose from No Value, Load from Keyfactor Secrets or Load From PAM Provider.

A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

CyberArk

Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

- Safe—The name of the safe the credential resides in.
- Object—The name of the username or password object in the safe.
- Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

This service account user needs appropriate permissions in the CA security settings to accomplish the tasks you plan to carry out for this CA through the Management Portal. For example:

- Certificate enrollment
- Certificate revocation
- Certificate key recovery
- Certificate request approval and denial

These tasks will be carried out on the CA in the context of the credentials you provide here. Access control for these tasks is controlled with Keyfactor Command security (see Security Roles and Claims on page 622) and the **Restrict Allowed Requesters** option, below.

Note: When this option is configured, enrollment and other tasks (e.g. revocation) are done in the context of the user configured here, not the user making the request in Keyfactor Command. This overrides the existing AD security policy used by Keyfactor Command.

Note: Once you have established explicit credentials to a forest for a CA, the forest will be included in the forest dropdown on the *Import Templates* dialog (see Certificate Templates on page 379).

Tip: The **Use Explicit Credentials** option is not needed if you are accessing your CA using a Keyfactor Universal Orchestrator. Enrollment is not supported when accessing a CA using an orchestrator, so the **Restrict Allowed Requesters** option is not relevant for this type of CA configuration.

The **Use Explicit Credentials** option is not used for EJBCA CAs.

Delegate Management Operations & Delegated Enrollment (DCOM-Microsoft CAs & CA Gateways)

Important: The Delegate options are only supported when using Active Directory as an identity provider.

The **Delegate Management Operations** and **Delegate Enrollment** boxes are used for CAs that support Windows integrated authentication to allow interactions with the CAs via Keyfactor Command to be done in the context of the user authenticated to Keyfactor Command using Kerberos authentication. These options also apply to users who authenticate to Keyfactor Command using Basic authentication, since Keyfactor Command performs pseudo delegation for these users. These options are not supported for users who authenticate using NTLM or Token authentication.

Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest

in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. If delegation is enabled, when a user authenticates with Kerberos or Basic authentication to Keyfactor Command, the Keyfactor Command server can delegate the user's credentials to the CA to provide end-to-end authentication without unpacking the credentials at the Keyfactor Command layer.

If you choose to disable one or both of the delegation options and have not enabled the *Use Explicit Credentials* option, interaction with the CA for the type of activity that is not delegated (e.g. management operations) is done in the context of the service account under which the Keyfactor API application pool is running. For more information, see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832.

Note: Use of explicit credentials is mutually exclusive of delegation.

Important: If you configure CA delegation and are using Kerberos authentication, you must also configure Kerberos constrained delegation for the CAs as per Configure Kerberos Constrained Delegation (Optional) on page 2824.

The types of interactions affected by these settings include:

- Approval of pending certificate requests (Delegate Management Operations)
- Denial of pending certificate requests (Delegate Management Operations)
- Revocation of certificates (Delegate Management Operations)
- Certificate key recovery (Delegate Management Operations)
- Certificate enrollment (Delegate Enrollment)

Note: If a workflow (see Workflow Definitions on page 230) is configured with a step that will result in a suspended state (e.g. pausing to wait for approvals) and the CA for the request is configured for delegation, the enrollment or revocation request made via the workflow will fail with an error indicating that the failure occurred because CA delegation is enabled. Workflows are not supported with CA delegation in the case where a suspended state may occur because it's possible that the initiating user's context may not be available all the way to the conclusion of the workflow.
When using workflow with steps that will result in a suspended state, do not use CA delegation. Instead, use the Keyfactor Command access control model provided by the **Restrict Allowed Requesters** option for enrollment (see Restrict Allowed Requesters (DCOM-Microsoft and HTTPS-EJBCA CAs) on the next page) and the Revoke permis-

sion for certificates at both the global and collection levels (see Certificate Collection Permissions on page 627).

If you choose to enable delegation, be aware that each user performing one of these delegable operations through the Management Portal must have the appropriate permissions to accomplish this task configured in the CA security settings.

Important: Granting users permissions in the CA security settings for certificate revocation, certificate key recovery, or certificate request approval and denial—e.g. the *Issue and Manage Certificates* permission—in order to support delegation of these operations through the Management Portal also grants these permissions to the users when operating outside the Keyfactor Command Management Portal. Any risk associated with this can be mitigated by implementing the Keyfactor Whitelist Policy Handler on each CA where such permissions are granted (see Installing the Keyfactor CA Policy Module Handlers on page 2846).

The **Delegate Management Operations** and **Delegate Enrollment** options are not used for EJBCA CAs.

Restrict Allowed Requesters (DCOM-Microsoft and HTTPS-EJBCA CAs)

The **Restrict Allowed Requesters** option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA. This option must be used to provide enrollment access if the **Delegate Enrollment** option is disabled. With this option checked, you must include at least one role in the ***Allowed Requester Security Roles*** table for enrollment to work.

This option is supported for all CAs, but it must be used for:

- Implementations using an identity provider other than Active Directory.
- Microsoft CAs where Windows integrated authentication is not supported.

  Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting *request certificates* for the selected security roles at the CA level on a Microsoft CA.
- EJBCA CAs.

The **Restrict Allowed Requesters** check box must be checked—and the **Allowed Requester Security Roles** populated—if the **Use Explicit Credentials** box is checked for a Microsoft CA that isn't accessed using integrated authentication.

Tip: For Microsoft CAs in a two-way trust environment you don't necessarily need to enable **Restrict Allowed Requesters** on the CA, though this may be required in some circumstances depending on the security configuration in the environment. However, templates for a two-way trust environment always require configuration of this option at a template level to support enrollment (see Certificate Template Operations on page 381).

In addition to granting permissions at the CA level using this option, you need enable the **Restrict Allowed Requesters** option to grant permissions on a template-by-template basis (see Certificate Templates on page 379).

Note: Access control for other types of interactions with the CA (e.g. revocation) is managed with standard security roles (e.g. the certificate revoke permission) at both the global and certificate collection level.

Authentication Certificate (HTTPS-EJBCA CAs)

Click the **Select Authentication Certificate** button to upload a client certificate in PKCS#12 format used to provide authentication to the EJBCA CA. This certificate is used to authenticate to the EJBCA database for synchronization, enrollment and management of certificates.

Note: Once you have established a connection to the EJBCA CA, it will be included in the forest dropdown on the *Import Templates* dialog (see Certificate Templates on page 379).

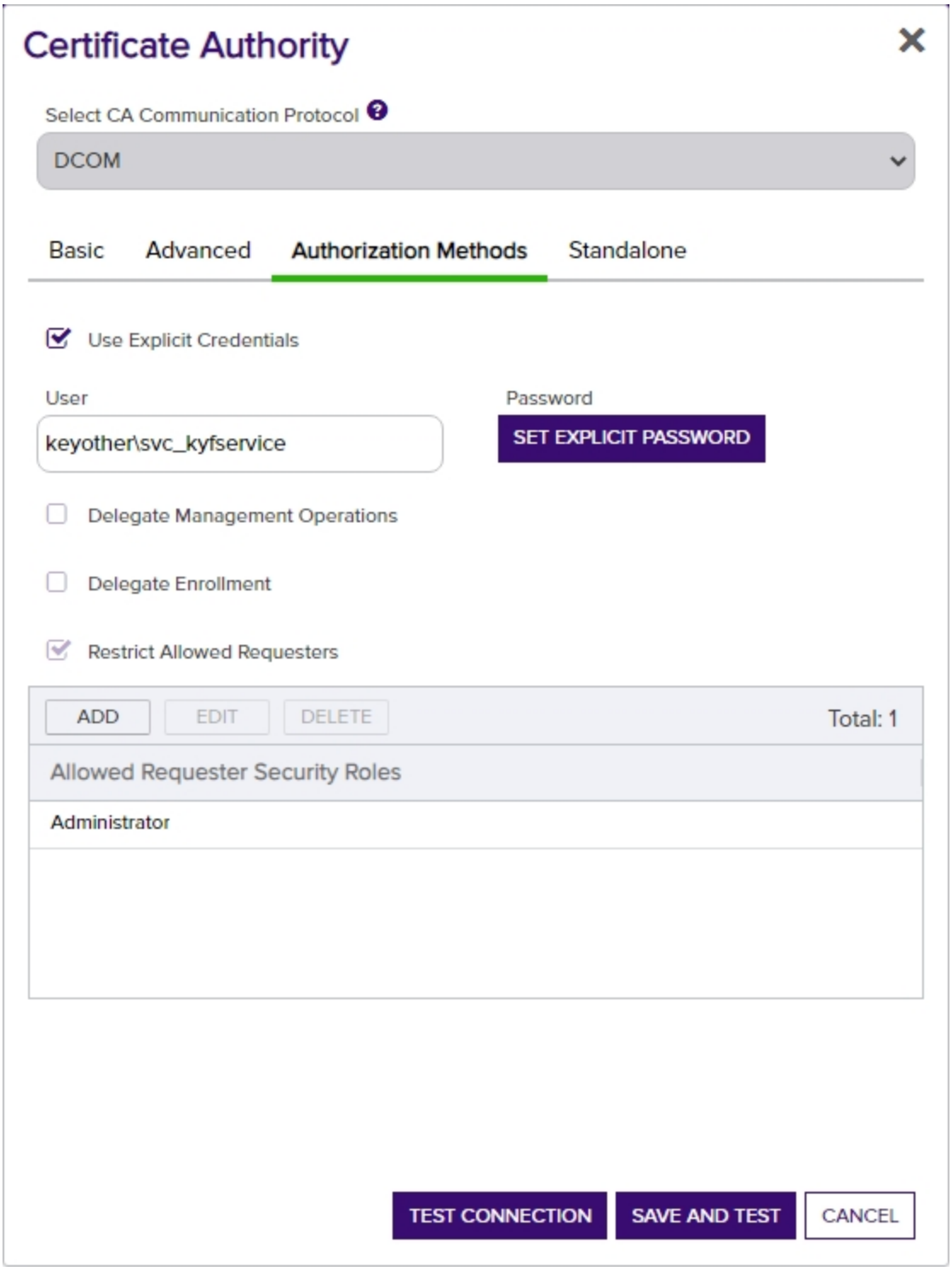

Certificate Authority
Select CA Communication Protocol
DCOM
Basic
Advanced
Authorization Methods
Standalone
Use Explicit Credentials
User
keyother\svc_kyfservice
Password
SET EXPLICIT PASSWORD
Delegate Management Operations
Delegate Enrollment
Restrict Allowed Requesters
ADD
EDIT
DELETE
Total: 1
Allowed Requester Security Roles
Administrator
TEST CONNECTION
SAVE AND TEST
CANCEL

Figure 220: Certificate Authority Authentication Methods Tab for a DCOM-Microsoft CA

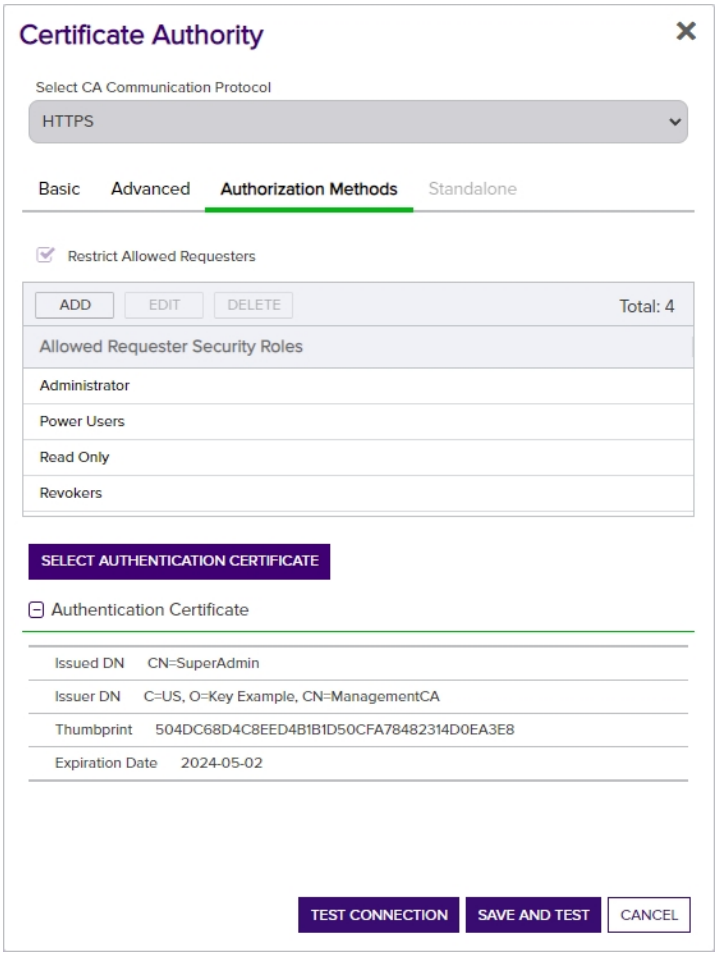


Figure 221: Certificate Authority Authentication Methods Tab for an HTTPS-EJBCA CA

Standalone Tab (Microsoft CAs)

To configure a standalone Microsoft CA, check the **Standalone** box.

Check the **Enforce RFC 2818 Compliance** box to require that certificate enrollments made through the Keyfactor Command Management Portal for this CA include at least one DNS SAN. This causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set.

If you have configured the CA for PFX enrollment on the Basic tab, the *Key Retention* field drop-down will display. Select a retention type. Enter the number of days, weeks, months, or years to keep the encrypted private key stored in the Keyfactor Command database based on the type

selected, then select the desired time frame (Day(s), Week(s), Month(s), or Year(s)). You will not have the option to choose a retention timeframe if you choose **Indefinite**.

Configuring private key retention allows the private keys for certificates enrolled through Keyfactor Command to be stored, encrypted, in the Keyfactor Command database for a user-definable period of time.

The private key retention configuration options are:

- Blank

  The private key will not be retained if the box is unchecked, or the *blank* option is selected.
- Indefinite

  The private key will be retained until it is explicitly deleted.
- After Expiration

  The private key will be retained until the specified number of days, weeks, months or years after the certificate expires, at which point it will be scheduled for deletion.
- From Issuance

  The private key will be retained until the specified number of days, weeks, months or years after the date on which the certificate was issued, at which point it will be scheduled for deletion.

Note: When the retention period is stored in the database, weeks are converted to 7 days, months are converted to 30 days, and years are converted to 365 days.

Tip: Setting the retention period to 0 will cause the private keys to be purged by the private key clean up job when it next runs, after the certificate expires or after the certificate is issued.

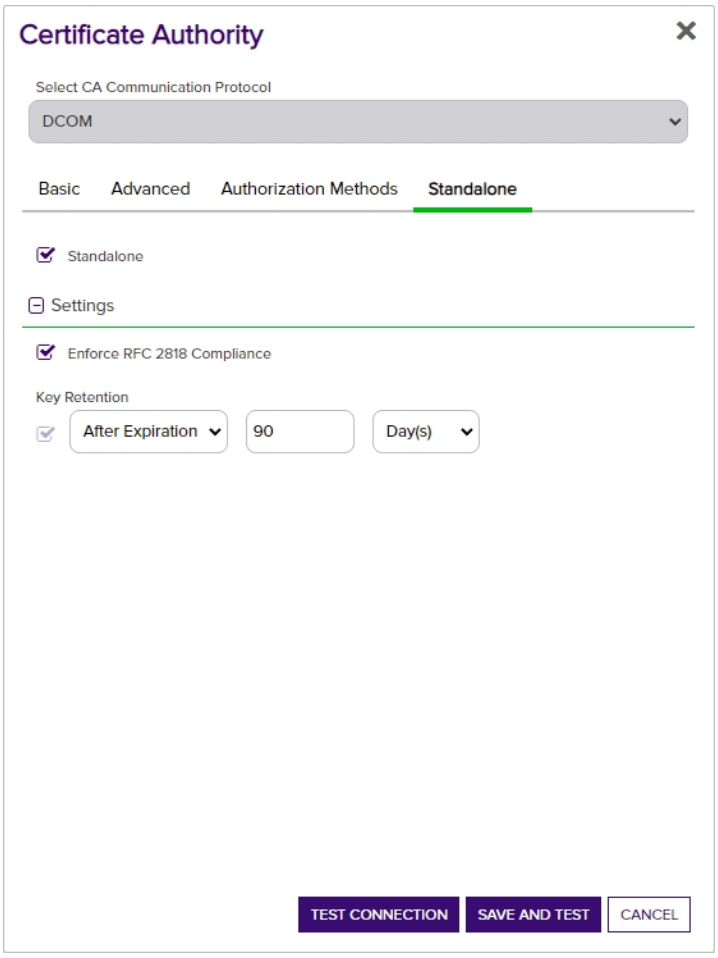


Figure 222: Certificate Authority Standalone Tab

5. Click **Test and Save** to add or update the CA, or click **Test Connection** to test the CA prior to saving (see Testing a CA Connection on page 353) for more information, including options for a failed CA test.

Tip: Once a CA record has been created for your CA, go to certificate templates (see Certificate Templates on page 379) and import templates for the CA. Template import is supported for both Microsoft and EJBCA CAs. Template import is not supported for the following:

- Non-domain-joined standalone Microsoft CAs (these don't use templates)
- CAs accessed via the Keyfactor Universal Orchestrator

Deleting a CA Record

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Agents > Management > Read
Certificate Authorities > Read
Certificate Authorities > Modify

To delete a CA record:

1. In the Management Portal, browse to *Locations* > *Certificate Authorities*.
2. On the Certificate Authorities grid, highlight the row in the CA grid and click **Delete** at the top of the grid or right-click the CA and choose **Delete** from the right-click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

A CA cannot be deleted if:

- It has scanning tasks enabled.
- It has certificates associated with it in the Keyfactor Command database.
- It is the last CA for its *Configuration Tenant* and there are certificate templates (see Certificate Templates on the next page) for that *Configuration Tenant* in Keyfactor Command.

Certificate Authority Monitoring

The two types of monitoring which Keyfactor Command offers for certificate authorities are configured on the Alert Recipients tab of the Certificate Authorities page at *Locations* > *Certificate Authorities*. Monitoring is not supported for CAs accessed with the Keyfactor Universal Orchestrator.

Certificate Authority Health Monitoring

Enable certificate authority health monitoring to receive email alerts when one or more of your CAs is not responding. Only CAs configured for synchronization will be monitored for health. To enable health monitoring, configure one or more recipients to receive the email messages and configure a health check schedule. You can choose to schedule the health checks either for daily at a specified time or at intervals of anywhere from every one minute to every 12 hours.

Certificate Authority Threshold Alerts

Enable threshold alerting to receive email alerts when a CA issues more or fewer certificates or experiences more failures or denials than configured for monitoring on the CA. Setting threshold monitoring is a two-step process:

1. Configure monitoring on the advanced tab (see Advanced Tab on page 365) for each CA.
2. Set the email recipients for the alerts on the alert recipients tab of the certificate authorities page.

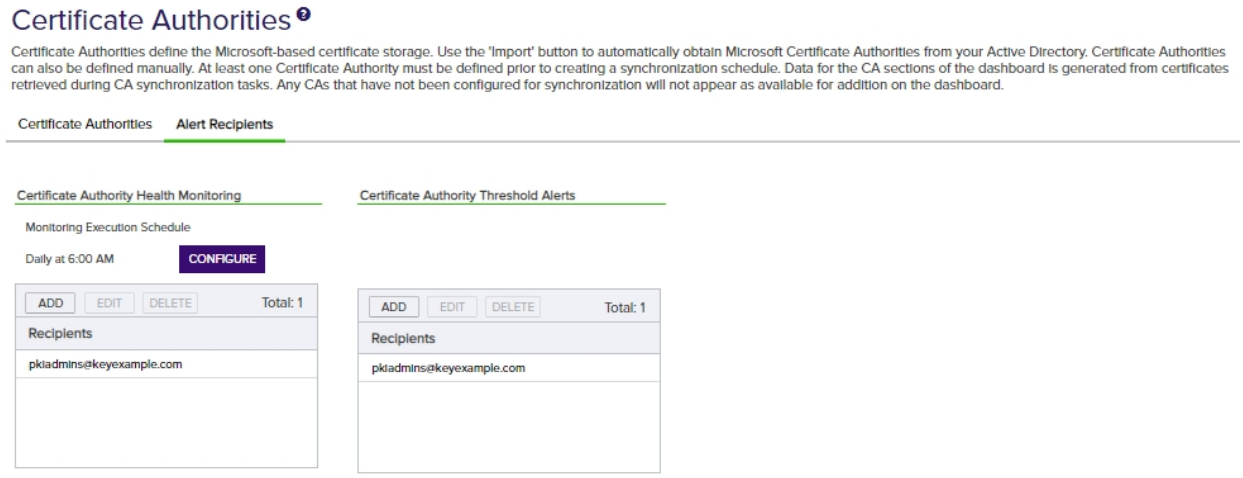


Figure 223: Certificate Authority Monitoring Recipients

2.1.8.2 Certificate Templates

During initial provisioning, the certificate templates in the primary Active Directory forest (the forest in which Keyfactor Command is installed) will be imported automatically by the Keyfactor Command configuration wizard. Templates for additional forests can be imported in a number of ways:

- For Microsoft CAs domain-joined to forests in a two-way trust with the primary forest, you can use the *Import Templates* option at any time.
- For Microsoft CAs domain-joined to forests in a one-way trust with the primary forest or to a forest having no trust with the primary forest, you can use the *Import Templates* option after you have configured a CA record for at least one Microsoft CA in the non-primary forest and enabled the *Use Explicit Credentials* option with credentials for the non-primary forest.
- For EJBCA CAs, you can use the *Import Templates* option after you have configured a CA record for at least one EJBCA CA.
- Templates that are associated with certificates that have been requested from a Microsoft CA in a forest other than the primary forest will appear in the templates grid as those certificates are synchronized to Keyfactor Command if you configure CA synchronization for the CA even if you don't use the import option.
- There's an automated process to import templates once every hour, on the hour. Templates are imported for Microsoft CAs in the primary forest, Microsoft CAs in any forests in a two-way trust with the primary forest, and any CAs that can be reached using the credentials configured in the CA record (the *Use Explicit Credentials* option for Microsoft CAs or the client certificate for EJBCA CAs). The automated template import only runs for CAs for which there is an active CA synchronization job configured. This automated sync is only enabled if the *Sync Templates* option on the **Service tab** of the Configuration Wizard is selected during installation (see Service Tab on page 2801 in the *Keyfactor Command Server Installation Guide*).

You will need to import templates if you add a new template or change the name or key size of a template after it has been imported into Keyfactor Command and don't want to wait for the automated import process or have not configured the automated process (see Importing Certificate Templates on the next page).

Note: When a template is imported into Keyfactor Command, a default template policy is added. Similarly, if a template is updated and no policy exists, a default policy is saved for it. The default policies have null values for everything, and will not override the system-wide policies.

Certificate templates need to be configured to support PFX and CSR enrollment (see Configuring Template Options on page 387).

Note: When EJBCA templates are imported, they are named using a naming scheme of:

- Short Name: *<end entity profile name>_<certificate profile name>*
- Display Name: *<end entity profile name>* (*<certificate profile name>*)

Only certificate profiles configured as *available* in a given end entity profile will be imported as templates associated with the given end entity profile name.

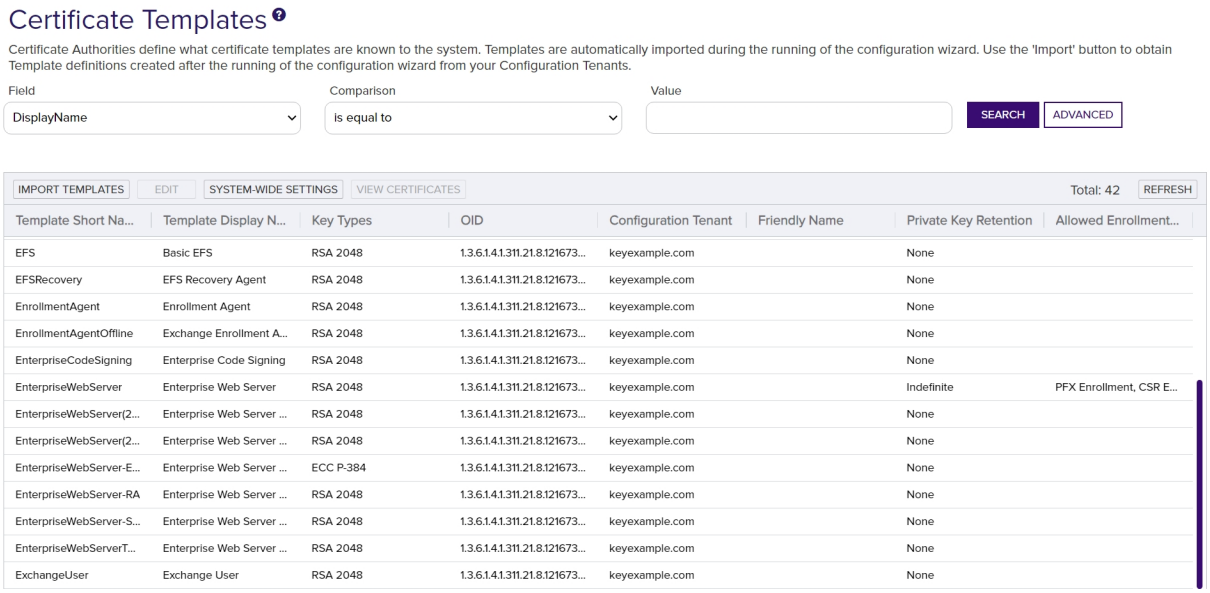


Figure 224: Certificate Templates

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Certificate Template Operations

Certificate templates are imported from their source rather than created in Keyfactor Command, which means there are limited operations that need to be performed in Keyfactor Command in relation to them. Supported actions on the certificate template page include:

- **Import Templates**

  The certificate templates in the primary Active Directory forest (the forest in which Keyfactor Command is installed) will be imported automatically by the Keyfactor Command configuration wizard during the Keyfactor Command installation if Keyfactor Command is installed on a domain-joined server. The template import option is used for templates from other sources, new templates created or edited after the Keyfactor Command installation, or template import for non-domain-joined Keyfactor Command servers.

- **Configure System-Wide Settings**

  The global settings option allows you to configure regular expressions, certificate subject defaults and policies that apply to all enrollments unless overridden by template-level settings.

- **Edit Template Options**

  Although templates are imported from their source, there are multiple Keyfactor Command-specific settings that can be configured on the templates to allow them to be used within the product.

- **View Certificates for a Template**

  The view certificates option takes you to the certificate search interface with the query field populated by the selected template.

Importing Certificate Templates

You only need to import templates if you have EJBCA CAs, Microsoft CAs in forests other than the forest in which Keyfactor Command was installed, are running Keyfactor Command on a non-domain-joined server, or have added a new template or changed the name or key size of a template after it has been imported into Keyfactor Command and don't want to wait for the automated import process or have not configured the automated process (see Certificate Templates on page 379).

To import certificate templates:

1. In the Management Portal, browse to *Locations* > *Certificate Templates*.
2. On the Certificate Templates page, click **Import Templates**.
3. In the Select Configuration Tenant dialog, select a configuration tenant in the dropdown.

   Tip: Previous versions of Keyfactor Command referred to the **Configuration Tenant** as the **Template Forest**.

   If you have a forest in a two-way trusted relationship with the forest in which Keyfactor Command is installed or have configured a Microsoft CA with the *Use Explicit Credentials* option or an EJBCA CA, the configuration tenant for this CA will appear in the dropdown. Import once

for each configuration tenant containing templates that you want to import. The import process may take several seconds.

Note: When EJBCA templates are imported, they are named using a naming scheme of:

- Short Name: *<end entity profile name>_<certificate profile name>*
- Display Name: *<end entity profile name>* (*<certificate profile name>*)

Only certificate profiles configured as *available* in a given end entity profile will be imported as templates associated with the given end entity profile name.

Tip: Out of the box, only templates for Microsoft CAs in the forest to which the Keyfactor Command server is joined and any Microsoft CAs in forests in a two-way trust with this forest can be imported using the template import. In order to import templates for other Microsoft CAs, you need to configure the *Use Explicit Credentials* option for each Microsoft CA for which you want to import templates and enter credentials valid for that CA with appropriate permissions to allow Keyfactor Command to query the CA for template records (see Authorization Methods Tab on page 367). Only one CA in each forest needs to be configured to allow the template import to function.

Configuring System-Wide Settings

System-wide settings apply to all enrollments done through Keyfactor Command unless they are overridden by template-specific settings (see Enrollment Regexes Tab on page 396, Enrollment Defaults Tab on page 398, and Policies Tab on page 399). Although system-wide settings are configured on the templates page, they also apply to enrollments done without a template (e.g. standalone CAs).

Note: System-wide settings replaced and enhanced selected application settings for enrollment beginning in release 10.

To configure system-wide options:

1. In the Keyfactor Command Management Portal, browse to *Locations > Certificate Templates*.
2. On the Certificate Templates page, click **System-Wide Settings** at the top of the grid.
3. When you open the system-wide settings, you will see three tabs. Configure the system-wide setting information with the appropriate data using the following instructions.
4. Click **Save** to save the system-wide settings. Click **Back** to return to the certificate templates page.

Enrollment RegExes Tab

Regular expressions for enrollment are used to validate that the data entered in the certificate subject fields meets certain criteria.

Tip: To use a system-wide enrollment regular expression and allow a specific template to bypass that regular expression, you can configure a template-level regular expression for the desired subject part and set it to nothing.

To configure a system-wide regular expression:

1. On the Enrollment RegExes tab, double-click a subject part row in the grid, right-click the row and choose **Edit** from the right-click menu, or highlight the row in the grid and click **Edit** at the top of the grid.
2. On the Enrollment RegEx dialog, in the *RegEx* field, enter a regular expression against which to validate the subject part. See Regular Expressions on page 402 for examples.
3. In the *Error* field, enter an error message to be displayed to the user in the enrollment pages of the Keyfactor Command Management Portal or as a response to an enrollment API request when the subject part referenced in the CSR or entered for a PFX does not match the regular expression defined for the subject part field. Note that the error message already includes the subject part followed by a colon (e.g. *Organization:* or *Invalid O provided*: depending on the interface). Your custom message follows this.
4. Click **Save** to save the regular expression.

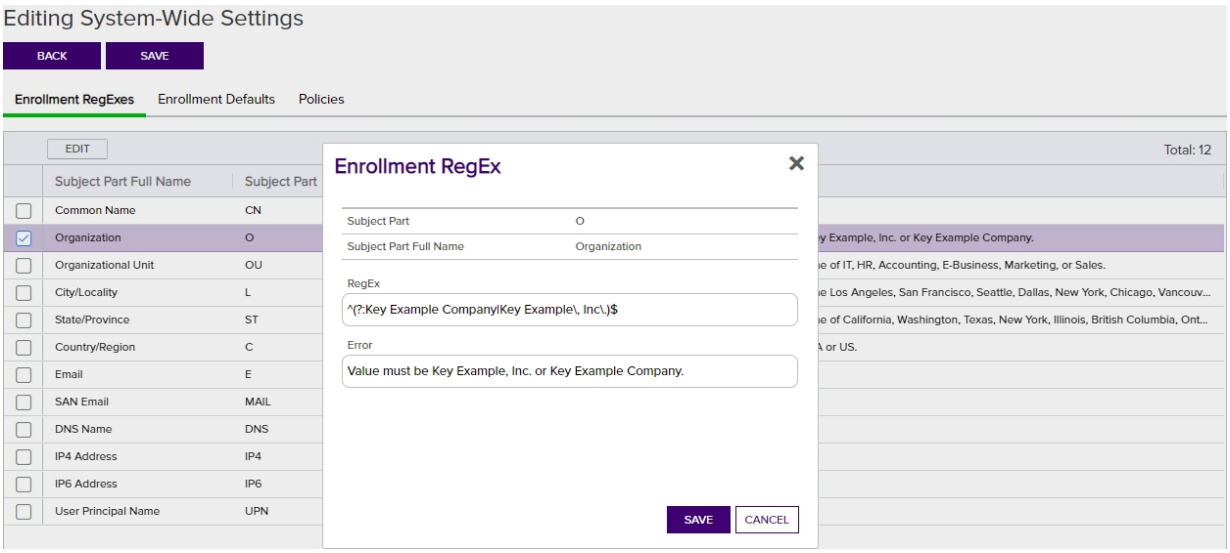


Figure 225: Configure System-Wide Enrollment Regular Expressions

Tip: To prevent users from adding a given SAN field to a certificate, create a regular expression on the field with the following value:

```
^$
```

This will disallow entry of any data in the SAN field and thus prevent users from submitting the certificate request with this SAN.

Enrollment Defaults Tab

Enrollment defaults allow you to define default values for select certificate subject parts that will auto-populate on the PFX enrollment and CSR generation pages in the Keyfactor Command Management Portal.

To configure a system-wide enrollment default:

1. On the Enrollment Defaults tab, double-click a subject part row in the enrollment defaults grid, right-click the row and choose **Edit** from the right-click menu, or highlight the row in the grid and click **Edit** at the top of the grid.

2. On the Enrollment Default dialog, in the *Value* field, enter a value to auto-populate in the PFX enrollment and CSR generation pages of the Keyfactor Command Management Portal. During PFX enrollment or CSR generation, the user can accept the value or modify it; it is not enforced.

3. Click **Save** to save the default.

Note: System-wide enrollment defaults do not apply to requests made with CSR enrollment or the Keyfactor API.

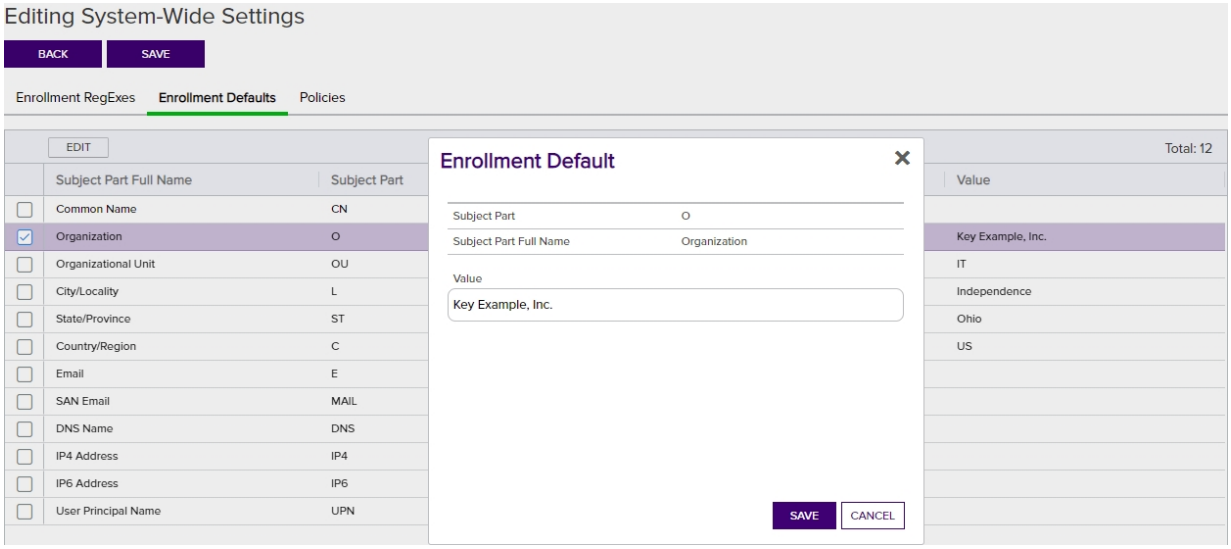


Figure 226: Configure System-Wide Enrollment Defaults

Tip: See also the *Subject Format* application setting, which takes precedence over enrollment defaults at both the system-wide and template level (see Application Settings: Enrollment Tab on page 609 in the *Keyfactor Command Reference Guide*).

Policies Tab

Note: When a template is imported into Keyfactor Command, a default template policy is added. Similarly, if a template is updated and no policy exists, a default policy is saved for it. The default policies have null values for everything, and will not override the system-wide policies.

Policies for templates cover the following settings:

Enrollment Policies

- Allow Wildcards

  Enable this option to allow certificates to be created containing wildcards (e.g. *.keyexample.com). The default is enabled.

- Allow Public Key Reuse

Enable this option to allow public keys to be reused on certificate renewals. The default is enabled.

- Enforce RFC 2818 Compliance

  Enable this option to force certificate enrollments made through Keyfactor Command to include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a DNS Name SAN, which will be set to *Read Only*. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. The default is disabled.

  Tip: For CA gateways, some cloud providers will automatically include SANs without you needing to enable the **Enforce RFC 2818 Compliance** option. Some cloud providers won't support submission of a SAN that matches the CN (which is the default when you enable the RFC 2818 option). Keyfactor recommends disabling this option for CA gateways.

Supported Key Types

- RSA Key Sizes

  A list of RSA key sizes that are valid for enrollment through Keyfactor Command. If a key size is not in this list, enrollment will not be supported for requests specifying that key size. To change the selected values, in the dropdown uncheck any values you do not wish to support. The default values are:

  ```
  2048, 3072, 4096
  ```

- ECC Curves

  A list of elliptic curve algorithms that are valid for enrollment through Keyfactor Command. To change the selected values, in the dropdown uncheck any values you do not wish to support. The default values are:

  ```
  P-256/prime256v1/secp256r1, P-384/secp384r1, P-521/secp521r1
  ```

- Allow Ed448 / Allow Ed25519

  Set global template values for allowing Ed448 and Ed25519 keys. Templates that utilize Ed448 or Ed25519 key types can be imported into Keyfactor Command. These key types are only available with EJBCA CAs. Default is disabled.

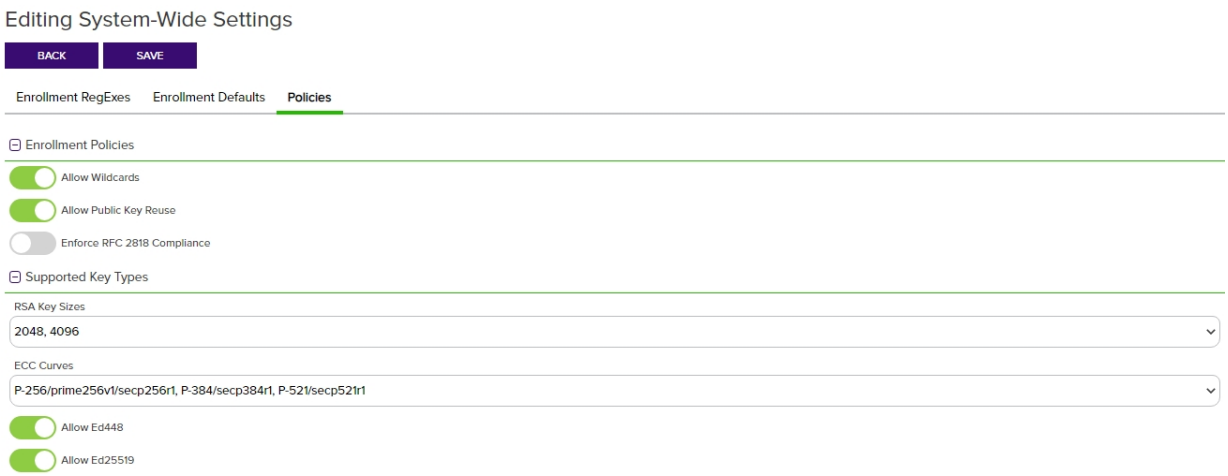


Figure 227: Configure System-Wide Policies

Configuring Template Options

The options configured in templates relate to how they appear and function for PFX and CSR enrollment in the Management Portal.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Collections > Metadata Modify
Certificate Templates > Read
Certificate Templates > Modify

To configure template options:

1. In the Keyfactor Command Management Portal, browse to *Locations > Certificate Templates*.
2. On the Certificate Templates page, double-click the template, right-click the template and choose **Edit** from the right-click menu, or highlight the row in the template grid and click **Edit** at the top of the grid.
3. When you open the certificate template for editing, you will see several tabs. Complete the template information with the appropriate data using the following instructions.
4. Click **Save** to save the changes to the template record. Click **Back** to return to the main certificate templates page without saving changes.

Details Tab

Details

The information in the *Details* section is for reference and cannot be edited. This includes:

- **Template Short Name**—The common name of the template. This name typically does not contain spaces.
- **Template Display Name**—The display name of the template.
- **Key Size**—The minimum supported key size of the template.
- **OID**—For a Microsoft certificate template, the object ID of the template retrieved from Active Directory. For an EJBCA certificate template, this field is generated within Keyfactor Command as an object identifier, but does not follow official OID conventions.
- **Curve**—For ECC templates, the elliptic curve algorithm defined for the certificate template.

Key Types and Sizes

The *Key Types and Sizes* section displays all the configured key types information for the template.

Friendly Name

In the *Friendly Name* section, enter a **Friendly Name**, if desired. Template friendly names, if configured, appear in template selection dropdowns in place of the template short names. This can be useful in environments where the template short names are long or not very human readable. This setting is not required to enable enrollment or configure private key retention.

Allowed Enrollment Types

In the *Allowed Enrollment Types* section, click the toggle buttons to enable the options for **CSR Enrollment**, **PFX Enrollment** and/or **CSR Generation** as desired. Enabling these options causes the template to appear in dropdowns in the corresponding section of the Management Portal. In the case of CSR Enrollment and PFX Enrollment, the templates only appear in dropdowns on the enrollment pages if they are available for enrollment from a CA also configured for enrollment within Keyfactor Command (see Adding or Modifying a CA Record on page 354).

If you wish to use *One-Click Renewal* for certificates, the **Allow One-Click Renewals** option must be enabled in both the templates and CAs to which you want *One-Click Renewal* to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354). For more information about one-click renewals, see Renew on page 69.

Private Key Retention

In the *Private Key Retention* section, click the toggle button to enable **Private Key Retention**, if desired, and select the retention type in the dropdown. Enter the number of days, weeks, months, or years to keep the encrypted private key stored in the Keyfactor Command database based on

the type selected, then select the desired time frame (Day(s), Week(s), Month(s), or Year(s)). You will not have the option to choose a retention timeframe if you choose **Indefinite**.

Configuring private key retention allows the private keys for certificates enrolled through Keyfactor Command to be stored, encrypted, in the Keyfactor Command database for a user-definable period of time.

Important: Any templates that are configured on the Microsoft CA Issuance Requirements tab for CA certificate manager approval cannot be used for PFX enrollment and associated pending, issued, and denied alerting in Keyfactor Command without configuring private key retention. Edit the template in Keyfactor Command, and on the Details tab check the Private Key Retention box. Set the dropdown to some value other than blank and for retention options of *After Expiration* or *From Issuance*, enter a value for the number of days, weeks, months or years to retain the private key. Without this setting, the template will not display on the template dropdown during PFX enrollment.

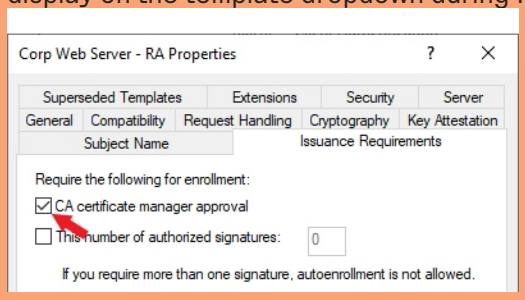


Figure 228: Microsoft Issuance Requirements on a Template for Manager Approval

Note: This does not apply to certificate requests requiring approval at a Keyfactor Command workflow level.

The private key retention configuration options are:

- Blank

  The private key will not be retained if the box is unchecked, or the *blank* option is selected.
- Indefinite

  The private key will be retained until it is explicitly deleted.
- After Expiration

  The private key will be retained until the specified number of days, weeks, months or years after the certificate expires, at which point it will be scheduled for deletion.
- From Issuance

The private key will be retained until the specified number of days, weeks, months or years after the date on which the certificate was issued, at which point it will be scheduled for deletion.

Note: When the retention period is stored in the database, weeks are converted to 7 days, months are converted to 30 days, and years are converted to 365 days.

Tip: Setting the retention period to 0 will cause the private keys to be purged by the private key clean up job when it next runs, after the certificate expires or after the certificate is issued.

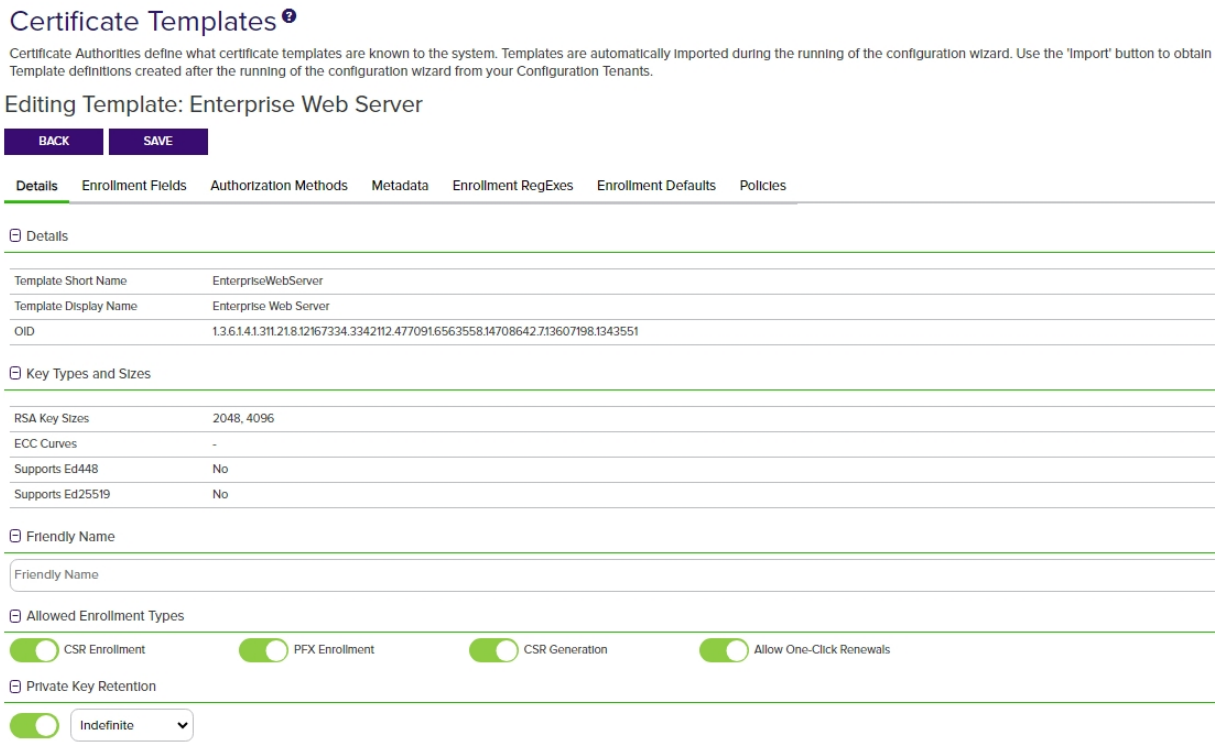


Figure 229: Certificate Template: Details Tab for a Microsoft Template

Enrollment Fields Tab

On the Enrollment Fields tab, you can add custom enrollment fields. These are configured on a per-template basis to allow you to submit custom fields with CSR enrollments and PFX enrollments to supply custom request attributes to the CA during the enrollment process. This functionality offers such benefits as:

- Preventing users from requesting invalid certificates, based on your specific certificate requirements per template.

- Providing additional information to the CA with the request.

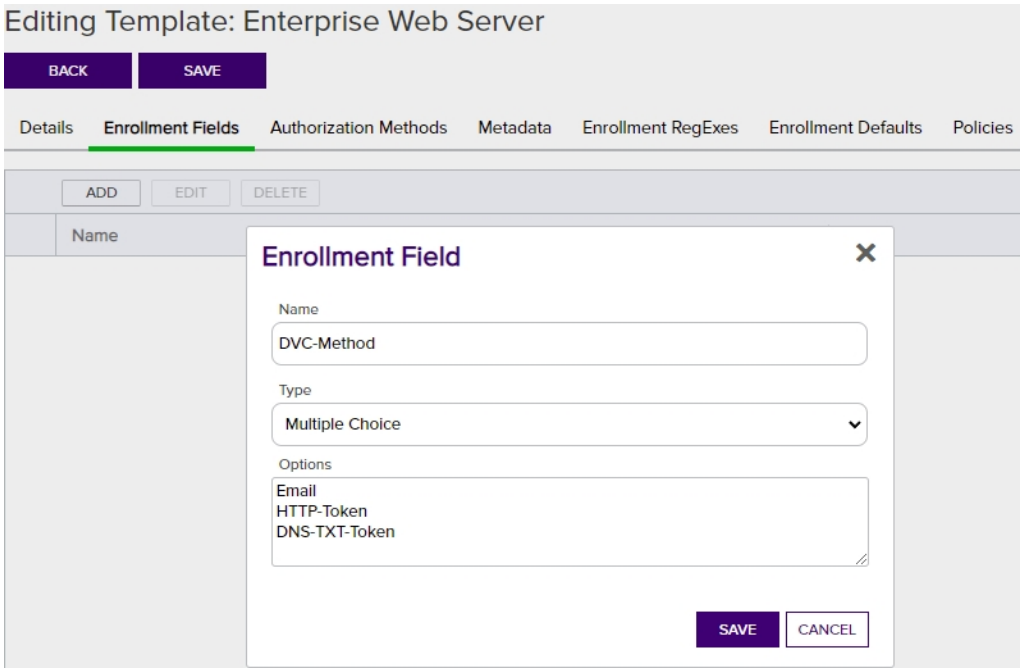


Figure 230: Configure Template: Enrollment Fields Tab

Once created on the template, these values are shown in Keyfactor Command on the PFX and CSR enrollment pages in the *Additional Enrollment Fields* section. The fields are mandatory during enrollment. The data will appear on the CA / Issued Certificates attribute tab for certificates enrolled with a template configured with Keyfactor Command enrollment fields.

Note: These are not metadata fields, so they are not stored in the Keyfactor Command database, but simply passed through to the CA. The CA in turn could, via a gateway or policy module, use this data to perform required actions.

On the Enrollment Fields tab you can add, edit and delete enrollment fields.

To add a new enrollment field:

1. On the **Enrollment Fields** tab of the selected template click **Add**. If there are existing fields configured they will appear in a list on this tab.
2. Enter a **Field Name** for the new custom field. This name will appear on the enrollment pages.

3. Select a **Parameter Type**. The options are:
    - **String**: A free-form data entry field.
    - **Multiple Choice**: Provides a list of acceptable values for the field. A text box will open up below this choice for you to enter the list of acceptable values. Add each value on a separate line. Click **OK** to close the box.
4. Click **Save** to save and close the add window.

Authorization Methods Tab

The **Restrict Allowed Requesters** option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command using this template. This is typically used for EJBCA templates and Microsoft templates that are not in the local Active Directory forest, since in these cases, Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates; this setting replaces that functionality. This setting is similar to setting *request certificates* for the selected security roles at the template level on a Microsoft CA. For multi-forest environments, this setting should be used on any templates from forests other than the Keyfactor Command forest that will be used for enrollment regardless of the type of trust between the forests, including two-way trusts.

Tip: In addition to granting permissions at the template level, you may need to enable the **Restrict Allowed Requesters** option to grant permissions at the CA level (see Adding or Modifying a CA Record on page 354). This is generally only required for untrusted CAs (including CAs in a forest with a one-way trust with the forest in which the Keyfactor Command server is located), but may be needed for CAs in a forest with a two-way trust with the Keyfactor Command forest depending on the security configuration in the environment.

On the Authorization Methods tab you can add, edit and delete allowed requesters.

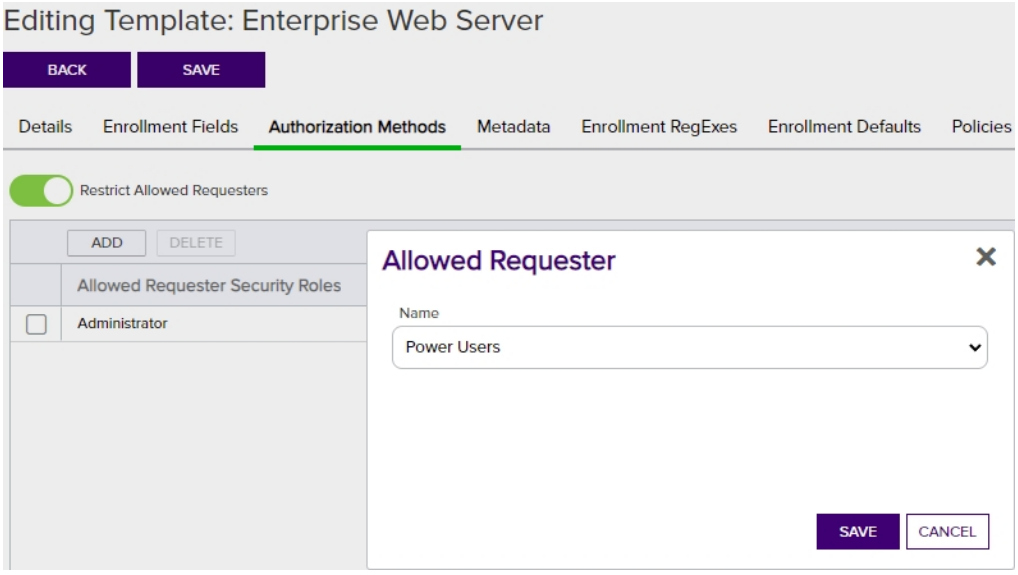


Figure 231: Certificate Template: Authorization Methods Tab

To add a new allowed requester, click to toggle the **Restrict Allowed Requesters** button and:

1. On the **Authorization Methods** tab of the selected template click **Add**. If there are existing requesters configured, they will appear in a list on this tab.
2. In the Security Role dropdown, select a Keyfactor Command security role (see Security Roles and Claims on page 622) to grant enrollment permissions on the template.
3. Click **Save** to save and close the add window.

Metadata Tab

From the **Metadata** tab you can:

- View the metadata field settings for that specific template.
- Configure how (or whether) the metadata fields will appear during enrollment for that specific template.

System-Wide Settings

System-wide metadata fields are defined in System Settings (see Certificate Metadata on page 710) and displayed here.

Template Settings

Once the system-wide metadata has been defined, the **Enrollment Handling** setting can be configured on a template-specific basis, potentially overriding a system-wide *required, hidden or optional* setting for *that metadata field* on *that template*, causing only the set of fields configured for the template to appear on the PFX and CSR enrollment pages when the template is selected, and determining if they are required or optional.

Tip: This allows an administrator to apply *required, hidden or optional* settings to a metadata field on a per-template basis so that only certain metadata fields appear on certain templates. For example, if metadata fields A and B are set to *required* or *optional* and Metadata field C is set to *hidden* for the WebServer template, only A and B will appear during enrollment with that template.

A default value for a metadata field can also be configured that is different from, and overrides, the default value entered for the system-wide metadata field. For string metadata fields, a regular expression validation and error message can also be configured on a template-specific basis. The order in which the metadata fields appear can be changed globally (see Sorting Metadata Fields on page 715).

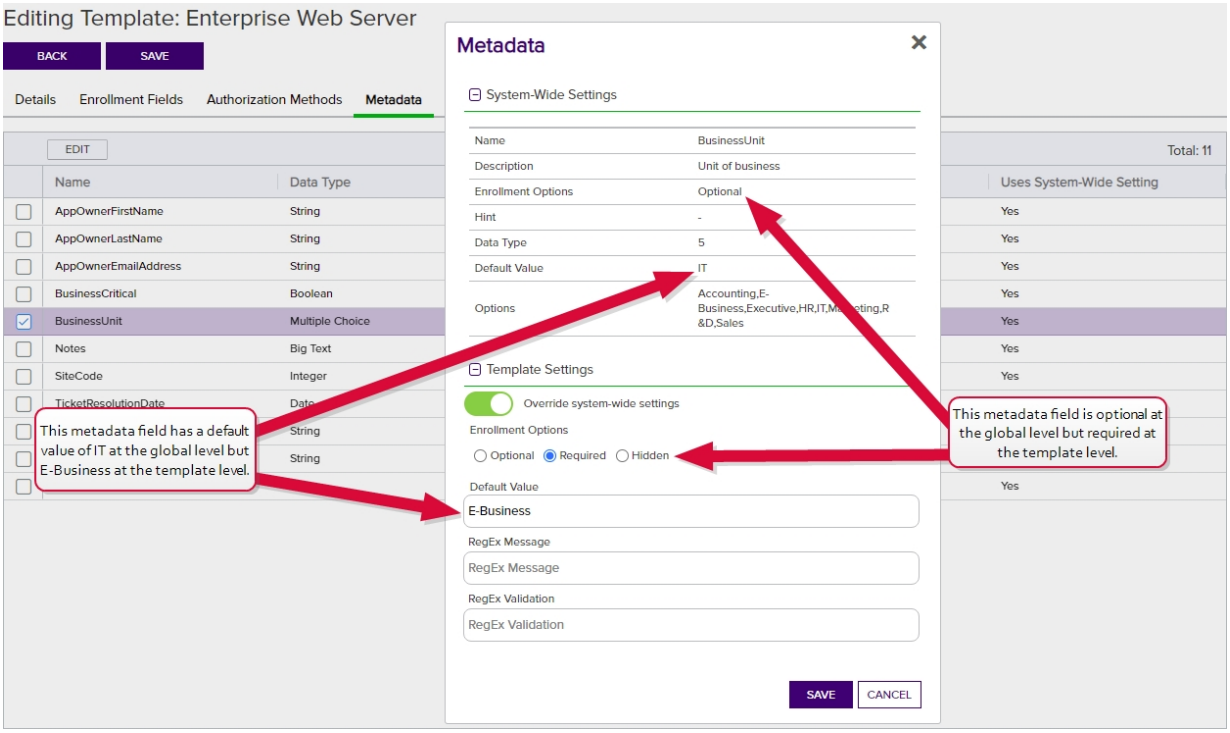


Figure 232: Certificate Template: Metadata Tab

The Metadata grid columns can be sorted by clicking the column heading (except Default Value). The columns are:

- **Name**: The name of the metadata field.
- **Data Type**: The metadata field type: *String, Integer, Date, Boolean, Multiple Choice, or Big Text*.
- **Enrollment Handling**: The handling of the metadata field during enrollment*: Optional, Required or Hidden*.
- **Default Value**: The default value during enrollment, if there is one, will be displayed.
- **Uses System-Wide Settings**: Displays *Yes* if system-wide settings are in effect for this template, or *No* if template-specifc settings are in effect.

To configure metadata fields for a template:

1. On the Metadata tab, double-click a row in the metadata grid, right-click the row and choose **Edit** from the right-click menu, or highlight the row in the grid and click **Edit** at the top of the grid.
2. In the Metadata dialog in the *System-Wide Settings* section, review the existing system-wide settings for the metadata field.
3. In the *Template Settings* section, click to toggle the **Override system-wide settings** button. Configure the template-level settings for the metadata field. The available fields will vary depending on the type of the metadata field and may include:
    - Choose the *Enrollment Options* for this template by selecting the appropriate radio button:
        a. **Optional**: The metadata field will appear during enrollment with this template, but it will not be required to complete enrollment.
        b. **Required**: This field will be required in order to complete enrollment with this template.
        c. **Hidden**: This field will not be displayed during enrollment with this template.
    - Set the *Default Value* if desired. If no default value is desired, the field may be left blank. For Multiple Choice type metadata fields, this field will appear as a dropdown where you can select from the existing values configured for the metadata field.
    - If desired, set a *RegEx Message* and *RegEx Validation* string specific to the template used to validate the value upon enrollment entry, and any error message to display if the entry does not match the regex definition. For more information, see Adding or Modifying a Metadata Field on page 710. This option is supported for string type metadata fields.
4. Click **Save** on the Metadata dialog to save changes for each metadata field.

Enrollment Regexes Tab

Enrollment Regexes can be applied at either the template-specifc level or system-wide level. Template-level regular expressions are used to validate that the certificate subject data entered on the CSR enrollment, CSR generation, and PFX enrollment pages meets certain criteria. Template-level regular expressions differ from system-wide regular expressions (see Configuring System-Wide Settings on page 382) as they apply on a per-template basis, rather than system-wide. In the case of a conflict in a regular expression between system-wide and template-level definitions, the template-level regular expression takes precedence.

Tip: To use a system-wide enrollment regular expression for a subject part and allow a specific template to bypass that regular expression, you can configure a template-level regular expression for the desired subject part and set it to no value.

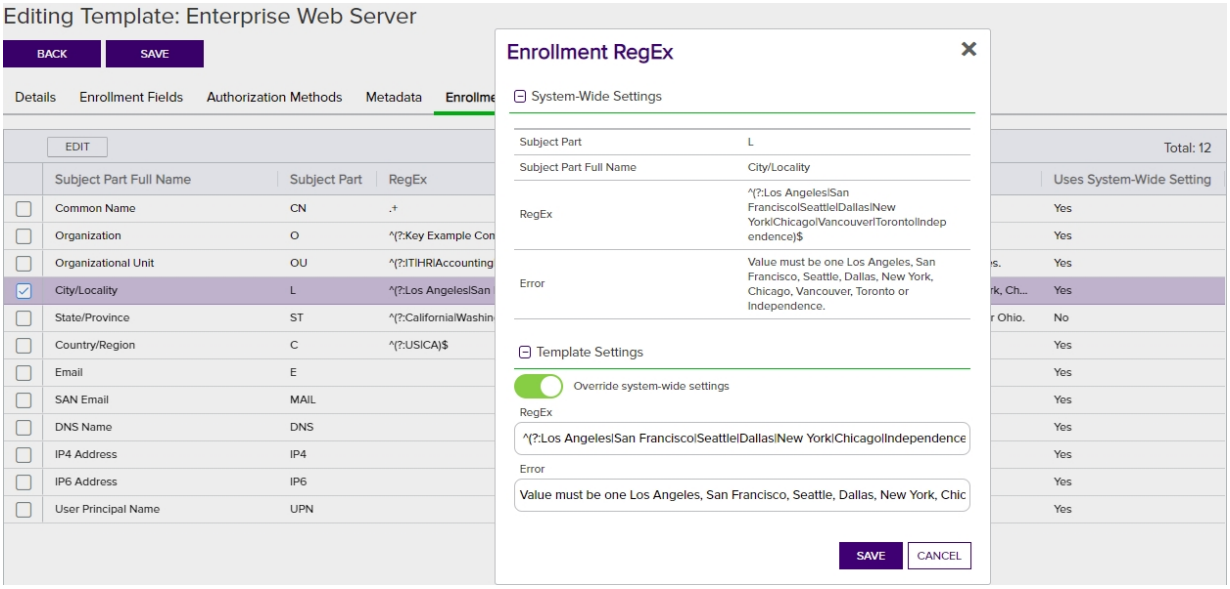


Figure 233: Certificate Template: Enrollment RegExesTab

The Enrollment Regexes grid columns can be sorted by clicking the column heading (except Regex and Error). The columns are:

- **Subject Part Full Name**: The descriptive name of the certificate subject part (e.g. Common Name).
- **Subject Part**: The code for the certificate subject information part. For instance, CN=Common Name.
- **RegEx**: The regular expression to apply to the subject part.
- **Error**: The error message to display (upon **Save** when enrolling), when the entry does not meet the specified criteria.

- **Uses System-Wide Settings**: Displays **Yes** if system-wide settings are in effect for this template, or *No* if template-specifc settings are in effect.

To configure template regular expression fields for a template:

1. On the Template Regexes tab, double-click a row in the regular expression grid, right-click the row and choose **Edit** from the right-click menu, or highlight the row in the grid and click **Edit** at the top of the grid.
2. In the Enrollment RegEx dialog in the *System-Wide Settings* section, review the existing system-wide settings for the subject part.
3. In the *Template Settings* section, click to toggle the **Override system-wide settings** button. Enter a regular expression in the **RegEx** field. See Regular Expressions on page 402 for examples.
4. In the **Error** field enter the error message to display during enrollment if the data entered for the subject part does not meet the validation rule.
5. Click **Save** on the Enrollment RegEx dialog to save each template-level regular expression.

The regular expressions will be applied at the time of enrollment. Entries which do not match the regular expression requirements will be flagged with an error message during enrollment entry when you click **Enroll** (or **Generate** for CSR generation).

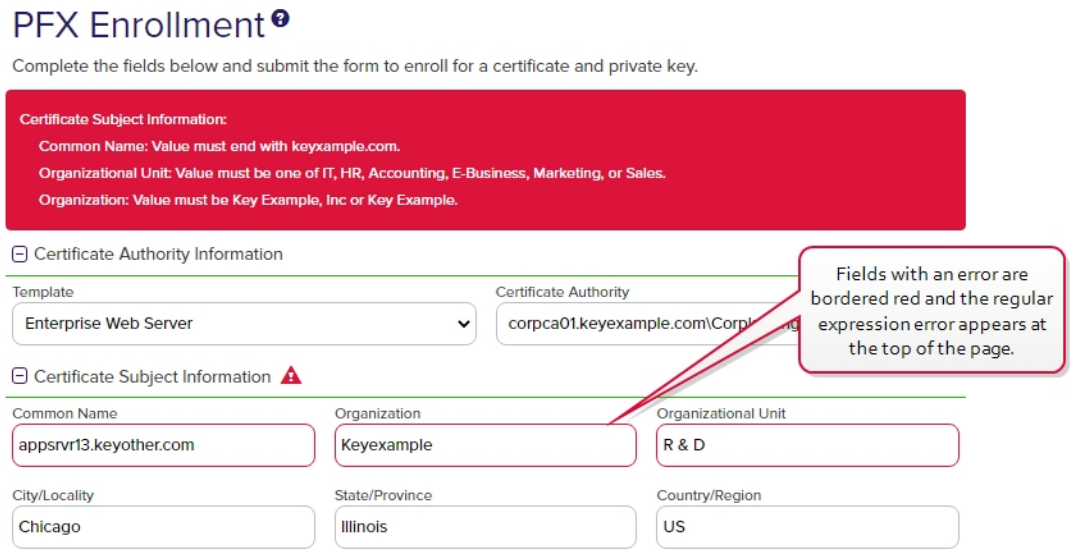


Figure 234: Certificate Template: Template Regular Expression Error on Enrollment

Tip: To prevent users from adding a given SAN field to a certificate, create a regular expression on the field with the following value:

^$

This will disallow entry of any data in the SAN field and thus prevent users from submitting the certificate request with this SAN.

Enrollment Defaults Tab

Template-level enrollment defaults allow you to define default values for certificate subject parts that will auto-populate on the PFX enrollment and CSR generation pages in the Keyfactor Command Management Portal. Template-level default values differ from system-wide default values (see Configuring System-Wide Settings on page 382) as they apply on a per-template basis, rather than system-wide. In the case of a conflict in a default value between system-wide and template-level definitions, the template-level default values takes precedence.

Note: These default values will not be applied to the additional SANs fields in CSR Enrollment.

Tip: To use a system-wide enrollment default value in a subject part and allow a specific template to bypass that default value, you can configure a template-level default value for the desired subject part and set it to no value.

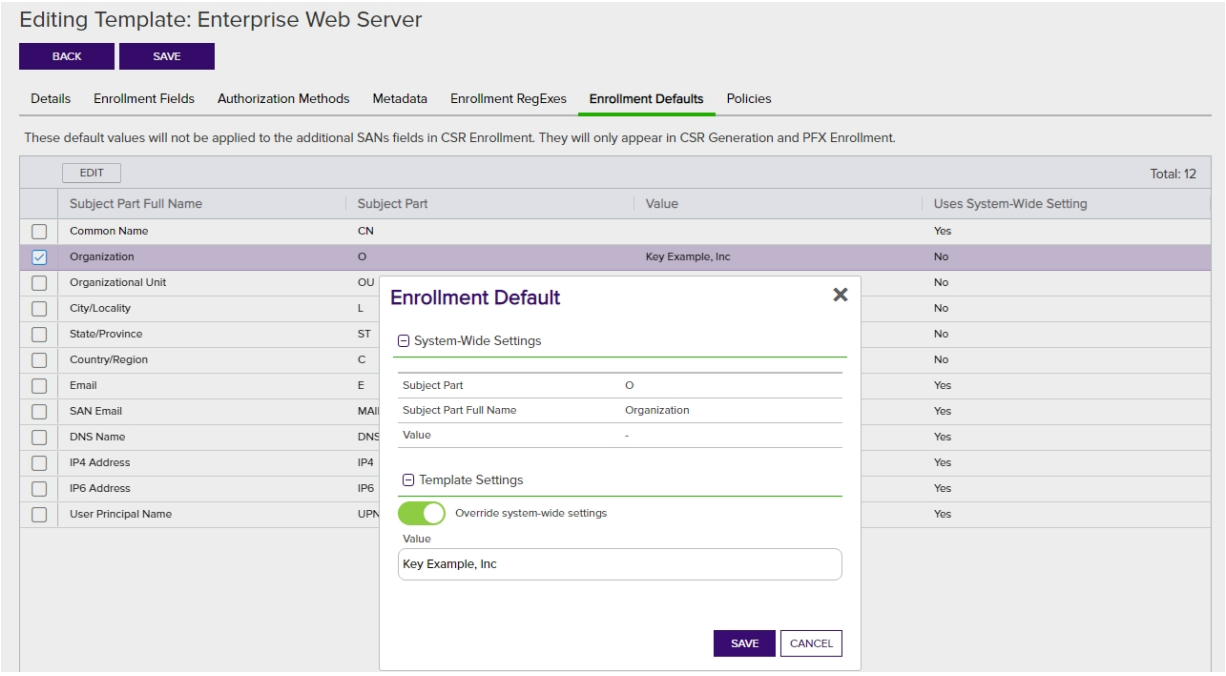


Figure 235: Certificate Template: Enrollment Defaults Tab

The Enrollment Defaults grid columns can be sorted by clicking the column heading (except Value). The columns are:

- **Subject Part Full Name**: The descriptive name of the certificate subject part (e.g. Common Name).
- **Subject Part**: The code for the certificate subject information part. For instance, CN=Common Name.
- **Value**: The default value to apply to the subject part.
- **Uses System-Wide Settings**: Displays **Yes** if system-wide settings are in effect for this template, or *No* if template-specific settings are in effect.

To configure template-level default values for a template:

1. On the Enrollment Defaults tab, double-click a row in the defaults grid, right-click the row and choose **Edit** from the right-click menu, or highlight the row in the grid and click **Edit** at the top of the grid.
2. In the Enrollment Default dialog in the *System-Wide Settings* section, review the existing system-wide default value for the subject part.
3. In the *Template Settings* section, click to toggle the **Override system-wide settings** button. Enter a template-level default value for the subject part in the **Value** field.
4. Click **Save** on the Enrollment Default dialog to save each template-level default.

Note: Enrollment defaults do not apply to requests made with CSR enrollment or the Keyfactor API.

Tip: See also the *Subject Format* application setting, which takes precedence over enrollment defaults at both the system-wide and template level (see Application Settings: Enrollment Tab on page 609 in the *Keyfactor Command Reference Guide*).

Policies Tab

The Policies Tab allows you to set template-level policy definitions which take precedence over system-wide settings (see Configuring System-Wide Settings on page 382).

Note: When a template is imported into Keyfactor Command, a default template policy is added. Similarly, if a template is updated and no policy exists, a default policy is saved for it. The default policies have null values for everything, and will not override the system-wide policies.

Enrollment Policies

The **Enrollment Policies** section displays the *System-Wide Setting* (*Yes* or *No*) for each of the template enrollment policies and allows you to **Override System-Wide Setting** for the specific template. Enabling **Override System-Wide Setting** will cause the system setting is to be disregarded and allow you to enable or disable the setting for that policy on the template. **Override System-Wide Setting** does not automatically set the policy to the opposite, the selection on the policy (enabled/disabled) will supersede any other settings.

- Allow Wildcards

  Enable this option to allow certificates to be created containing wildcards (e.g. *.keyexample.com) using this template.

- Allow Public Key Reuse

  Enable this option to allow private keys to be reused on certificate renewals made using this template.

- Enforce RFC 2818 Compliance

  Enable this option to force certificate enrollments made through Keyfactor Command for this template to include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set.

  Tip: For CA gateways, some cloud providers will automatically include SANs without you needing to enable the **Enforce RFC 2818 Compliance** option. Some cloud providers won't support submission of a SAN that matches the CN (which is the default when you enable the RFC 2818 option). Keyfactor recommends disabling this option for CA gateways.

Supported Key Types

The **Supported Key Types** section displays the *System-Wide Setting* (*value* or *Yes*/*No*) for the supported key type and allows you to Override System-Wide Setting for the specific template. Enabling **Override System-Wide Setting** will cause the system-wide setting to be disregarded, enable the settings field, and allow you to select the setting for that policy on the template. **Override System-Wide Setting** does not automatically set the policy to the opposite, the selection on the policy (values or enabled/disabled) will supersede any other settings.

Note: The supported key algorithms for a certificate template are determined based on the key size(s), type(s), and curve(s) returned for the template from the CA and the Keyfactor Command template policy settings at both the system-wide and template-specific levels.

Evaluation using both the CA configurations and the Keyfactor Command policy allows for a key size to be set on a template in Keyfactor Command for validation purposes that can be different than the CA template key size setting (perhaps the CA supports RSA 2048, RSA 4096, and ECC 256 for the template but the Keyfactor Command template-specific policy is set to disallow RSA and allow only ECC 256).
When configuring template-level policies for key information, only algorithms, sizes, and curves supported for the template by the CA will appear as options for configuration.

Depending on the template selected, one or more of these settings will be available for configuration:

RSA Key Sizes

A list of RSA key sizes that are valid for enrollment through Keyfactor Command for this template. If a key size is not in this list, enrollment will not be supported for requests specifying that key size. To change the selected values, in the dropdown check any values you wish to support. The values supported by Keyfactor Command are: 2048, 3072, 4096, 6144, 8192, and 16384. Only values that are valid for the template on the CA will appear as available for selection.

ECC Curves

A list of elliptic curve algorithms that are valid for enrollment through Keyfactor Command for this template. To change the selected values, in the dropdown check any values you wish to support. The values supported by Keyfactor Command are: P-256/prime256v1/secp256r1, P-384/secp384r1, and P-521/secp521r1. Only values that are valid for the template on the CA will appear as available for selection.

Allow Ed448 for Template / Allow Ed25515 for Template

Enable or disable support for Ed448 or Ed25519 keys on the template.

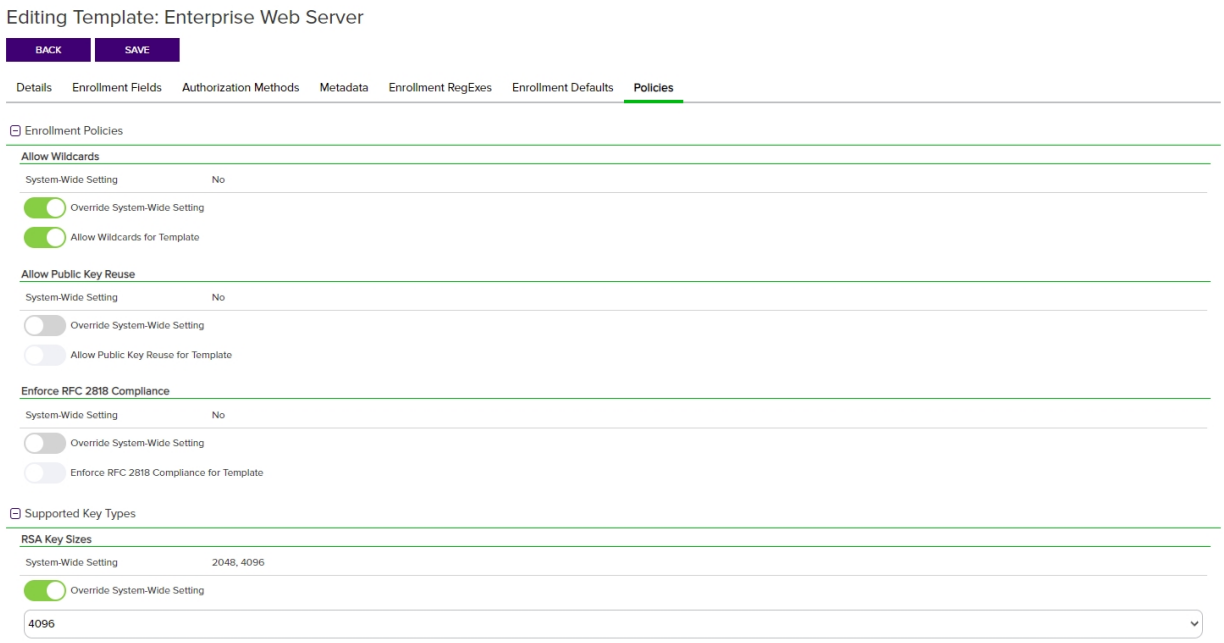


Figure 236: Certificate Template: Policies Tab

Tip: Templates that are configured for CA-level key archiving are not supported for enrollment done through Keyfactor Command. For a Microsoft CA, this is the "Archive subject's encryption private key" setting on the template. For an EJBCA CA, this is the "Key Recoverable" setting on the end entity profile, which only appears if key recovery has been enabled in system configuration. An error similar to the following on enrollment is an indication that a Microsoft template is configured to archive the private key:

```
The request is missing a required private key for archival by the server.
```

For enrollments done through Keyfactor Command for either Microsoft or EJBCA CAs, use Keyfactor Command private key retention (see Details Tab on page 387).

Viewing Template Certificates

To view the certificates in the Keyfactor Command database for a given template, highlight the template in the grid and click **View Certificates** at the top of the grid or right-click the template and choose **View Certificates** from the right-click menu. This will take you to the certificate search page with the query field populated by the selected template (see Certificate Search Page on page 34). You can save the search as a certificate collection at that point if desired (see Saving Search Criteria as a Collection on page 42).

Regular Expressions

Several fields on the CSR enrollment, CSR generation, and PFX enrollment pages support using regular expressions to validate that the data entered in the fields meets certain criteria. Both

certificate subject fields and metadata string fields can be configured with regular expressions. The certificate subject fields that support regular expressions are shown in Table 13: Supported Regular Expressions for Enrollment with Examples.

Regular expressions for enrollment can be defined at a global level to apply to all enrollments and at a template level to apply only to enrollments done with that template. Template-level definitions take precedence over global definitions.

Both the regular expressions that do the validation and the error message that the user receives when the validation fails are user definable. For example, for the common name field you could define a regular expression similar to the following:

```
^[a-zA-Z0-9'_\.\-]*\.keyexample\.com$
```

This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*. Using this regular expression would prevent users from requesting certificates with common names such as myserver.contoso.com, forcing them to request certificates for domain names that are valid for your organization. Your error message to the user in this case might be something like:

```
Common names must end with keyexample.com.
```

The error message to the user appears immediately once the user leaves the field being validated after entering data that doesn't meet the regular expression requirements.

Table 13: Supported Regular Expressions for Enrollment with Examples

Subject Part	Example

CN (Common Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_\.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least one character in the Common Name field in the enrollment pages.

O (Organization)	This regular expression requires that the organization name entered in the field be one of "Key Example Inc", "Key Example" or "Key Example Inc.": `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$`

Subject Part	Example
	The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “.keyexample1.com” or “.keyexample2.com”: `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly “130.101.” followed by anywhere between 1 and 3 numbers followed by exactly “.” followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$` This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:

Subject Part	Example
	`^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`
IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
MAIL (Subject Alternative Name: Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
UPN (Subject Alternative Name: User Principal Name)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`

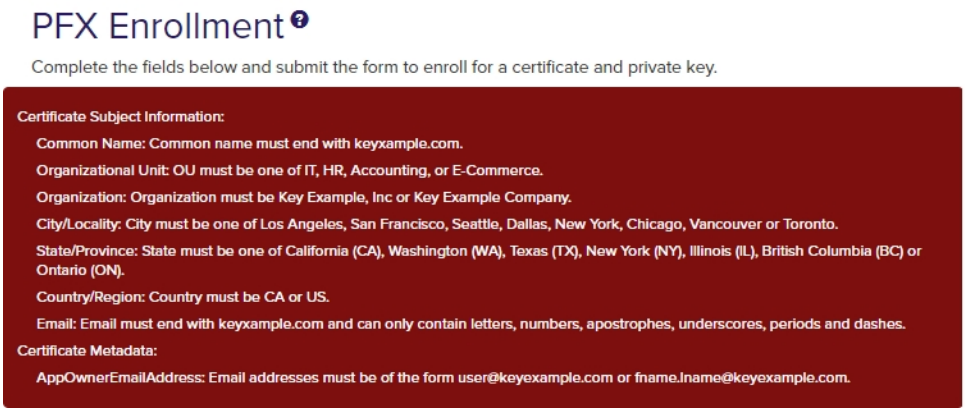


Figure 237: PFX Enrollment Regular Expression Validation Error

For more information about configuring regular expressions on metadata fields, see Certificate Metadata on page 710.

Using the Template Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

DisplayName

Complete or partial matches with the name of the template Display Name.

AllowedEnrollmentType

Complete or partial matches with allowed enrollment types on the template.

IsDefaultTemplate

The template is one of the Microsoft default templates (true/false). This is helpful to filter out the templates that you did/didn't create.

ConfigurationTenant

Complete or partial matches with the Configuration Tenant name.

FriendlyName

Complete or partial matches with the Keyfactor Command friendly name of the template.

ShortName

Complete or partial matches with the template Short Name.

HasPrivateKeyRetention

Private Key Retention is selected for this template (true/false).

ForestRoot

Complete or partial matches with the forest location.

Note: This will be deprecated in a future release and replaced with ConfigurationTenant.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

2.1.8.3 Certificate Stores

The certificate store feature in Keyfactor Command allows you to search for and inventory certificates from multiple types of certificate stores, import the certificates found in them into the Keyfactor Command database, add new certificates to the stores, and remove certificates from them. This feature uses Keyfactor orchestrators to communicate with the Keyfactor Command server. This section of the documentation describes the management tasks that can be done through the Management Portal. For information about installing and configuring the Keyfactor Universal Orchestrator, see the Installing Orchestrators on page 2875 guide.

Certificate stores are managed by configuring the store locations through the Management Portal, assigning an inventory schedule, and optionally assigning stores to containers (groups) for ease of management. You can create records for stores in the Management Portal manually or by using the discovery feature. Not all certificate store types support discovery; check the details of the certificate store types or any custom-built extensions you're using to determine whether discovery is supported.

Managing certificate stores requires that an appropriate instance of a Keyfactor orchestrator is running in the environment and has been approved in the Management Portal (see Orchestrator Management on page 496). Java and PEM certificate stores can be managed with an instance of the Keyfactor Java Agent running on the machine where the Java and PEM certificate stores are located or with the Keyfactor Universal Orchestrator and the Keyfactor Remote File extension. Amazon Web Services (AWS), F5, Citrix/NetScaler, Windows (IIS) certificate stores and more can be managed with the Keyfactor Universal Orchestrator and an appropriate Keyfactor custom-built extension. Keyfactor offers many custom-built extensions for the Keyfactor Universal Orchestrator on GitHub:

https://keyfactor.github.io/integrations-catalog/content/orchestrator

Some packages that may be of special interest to long-term users of Keyfactor Command are:

- AWS Certificate Store Manager
- Citrix NetScaler Certificate Store Manager
- F5 Certificate Store Manager
- IIS Certificate Store Manager
- Remote File Certificate Store Management (Java Keystores, PKCS12 files, PEM files, DER files, IBM Key Database files)

Once your certificate stores have been inventoried and their certificates imported into Keyfactor Command, you can use the standard Management Portal features for managing certificates—such as Expiration Alerts (see Expiration Alerts on page 167)—to manage the certificates from the certificate store locations even if the certificates were not generated by your Keyfactor Command configured CAs.

Most certificate store types can use **Privileged Access Management (PAM)** or **Keyfactor Secrets** to manage passwords for the servers or devices on which the certificates stores are located and on the certificate stores themselves, where applicable.

F5 and IIS Certificate Store Terminology

This section uses the following terminology for F5 and IIS certificate stores:

F5 CA Bundles REST

Certificates and keys for the *F5 CA Bundles REST* are those found within F5 *Bundles*. Note that the *ca-bundle* cannot be managed with Keyfactor Command, as it is protected and managed directly by F5. Only the *Include Bundles* may be managed with this option. This option uses the F5 iControl REST API. It is intended to be used with BIG-IP versions 13 and later. The F5 CA Bundles REST option supports certificate discovery on the F5 device and F5 high availability.

F5 SSL Profiles

Certificates and keys for the *F5 SSL Profiles* are those used by any applications configured for use by the F5 device. These are certificates that are available in the F5 interface as the SSL certificate list. This option uses the F5 SOAP API. It is intended to be used with BIG-IP version 12.

F5 SSL Profiles REST

Certificates and keys for the *F5 SSL Profiles REST* are those used by any applications configured for use by the F5 device. These are certificates that are available in the F5 interface as the SSL certificate list. This option uses the F5 iControl REST API. It is intended to be used with BIG-IP versions 13 and later. The REST version of F5 SSL Profiles supports certificate discovery on the F5 device and F5 high availability.

F5 Web Server

F5 Web Server REST

Certificates and keys for the *F5 Web Server REST* are those used by the device itself for the F5 portal and the API. This certificate is referred to as the *device certificate* within the F5 interface. This option uses the F5 iControl REST API. It is intended to be used with BIG-IP versions 13 and later. The F5 Web Server REST option supports F5 high availability.

IIS Revoked

The Untrusted Certificates store of the local computer.

IIS Trusted Roots

The Trusted Root Certification Authorities store of the local computer.

IIS Personal

The Personal store of the local computer.

Certificates and keys for the *F5 Web Server* are those used by the device itself for the F5 portal and the SOAP API. This certificate is referred to as the *device certificate* within the F5 interface. This option uses the F5 SOAP API. It is intended to be used with BIG-IP version 12.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Using the Certificate Store Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

AgentAvailable

Orchestrator has been approved and made available to manage certificate store jobs (true/false).

AgentID

Orchestrator Id matches or doesn't match the entered GUID (primarily used for internally generated searches when the user is redirected here from another page).

Category

Certificate store matches or doesn't match the referenced category. Categories include (plus categories from custom certificate store types you've entered):

Container

Complete or partial matches with one or more certificate store containers.

HasInventoryScheduled

Certificate store has an inventory job scheduled (true/false).

StorePath

Complete or partial matches with the full path to a certificate store—e.g. /opt/application/mystore.crt or c:\program files\application\mystore.jks.

- Amazon Web Services
- F5 CA Bundles REST
- F5 SSL Profiles
- F5 SSL Profiles REST
- F5 Web Server
- F5 Web Server REST
- File Transfer Protocol
- IIS Personal
- IIS Revoked
- IIS Roots
- Java Keystore
- NetScaler
- PEM File

ClientMachine

Complete or partial matches with the client machine(s) on which a store or stores may be found.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

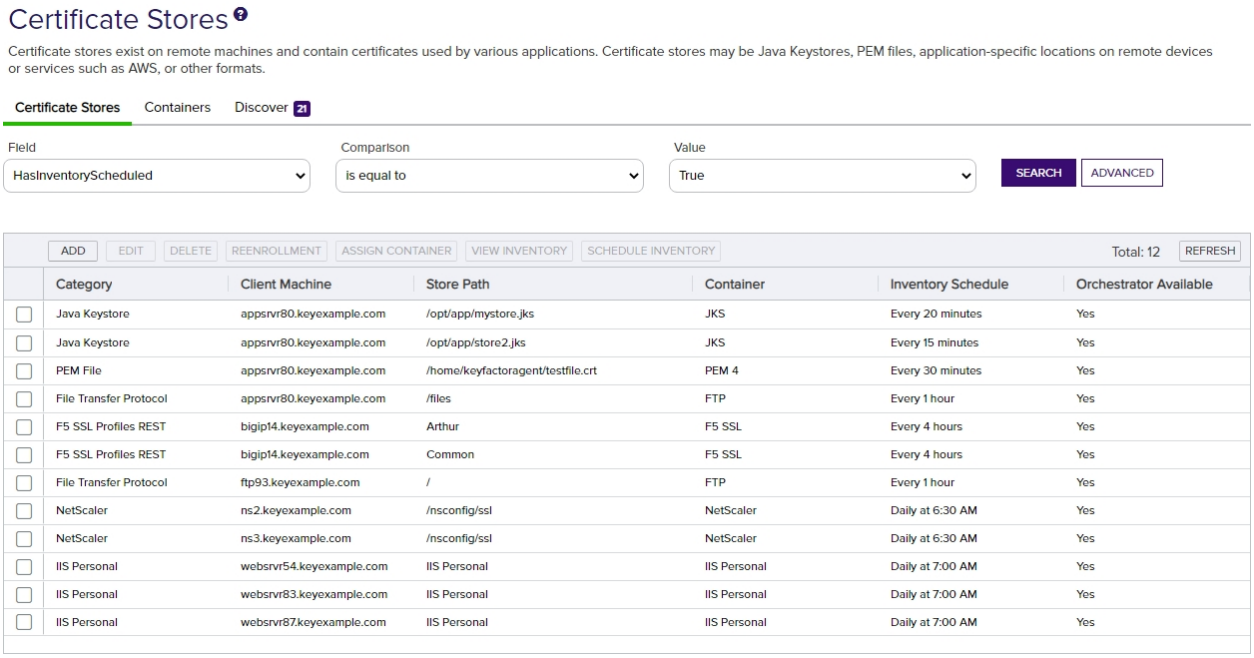


Figure 238: Simple Certificate Store Search

The search results can be sorted by clicking on a column header in the results grid for every column except Inventory Schedule and Orchestrator Available. Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may be adjusted by click-holding and dragging the line separating two column headers.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

Certificate Store Operations

To select a single row in the certificate store grid, click to highlight it and then select an operation from either the top of the grid or the right-click menu. The delete, schedule inventory and assign container operations can be done on multiple certificate stores at once. To select multiple rows, click the checkbox for each row on which you would like to perform an operation. Then select an operation from the top of the grid. The selected stores must all be of the same category (e.g. PEM or Java) to perform the assign container operation. The right-click menu supports operations on only one store at a time.

Adding or Modifying a Certificate Store

Before creating a certificate store in Keyfactor Command, you must approve an orchestrator to handle the store (see Approving or Disapproving Orchestrators on page 500).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Agents > Management > Read
Certificate Stores > Read
Certificate Stores > Modify

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Certificate stores can be added manually or, for some types of stores, automatically using a discover process (see Certificate Store Discovery on page 437).

Note: A user with the appropriate permissions may create more than one certificate store in a given location provided the stores are of different categories/types. Stores of the same type should still fail to be saved using the same target.

To define a new certificate store location manually or edit an existing one:

1. In the Management Portal, browse to *Locations* > *Certificate Stores*.
2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).
3. On the Certificate Stores tab, click **Add** to create a new store location, or click **Edit** from either the top or right-click menu to modify an existing one.
4. In the Certificate Stores dialog, select the type of certificate store in the **Category** dropdown. The values that appear here are the display names for the built-in certificate store types and any custom certificate store types you entered to match your custom extensions. Examples of Keyfactor-provided custom-built extensions on GitHub include:

    - AWS Certificate Store Manager
    - Citrix NetScaler Certificate Store Manager
    - F5 Certificate Store Manager
    - IIS Certificate Store Manager
    - Remote File Certificate Store Management (Java Keystores, PKCS12 files, PEM files, DER files, IBM Key Database files)

    Note: This field cannot be modified on an edit.

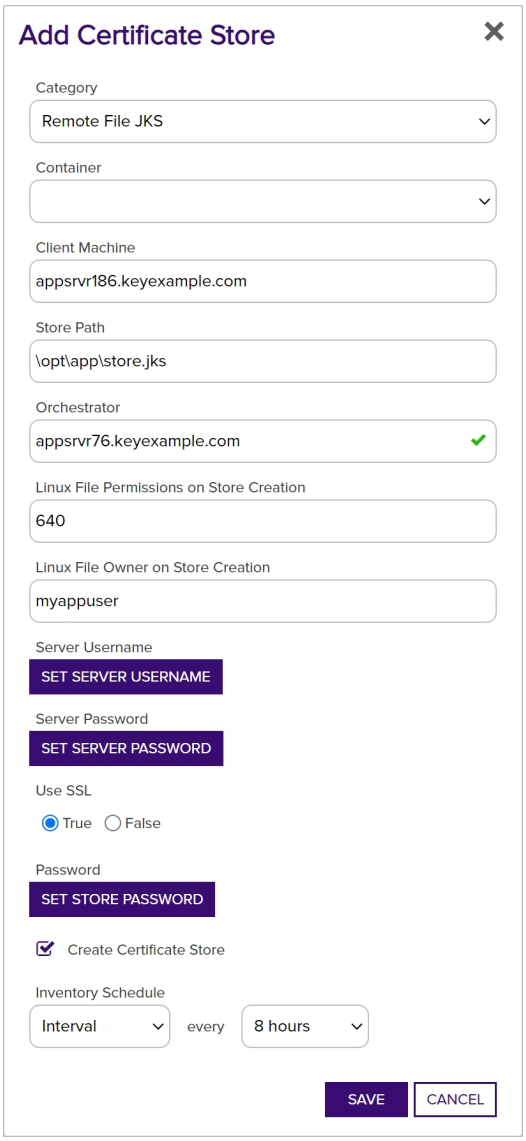


Figure 239: Add a Remote JKS Certificate Store

5. In the **Container** field, select a container into which to place the store for organization from your previously defined list, if desired. This field is optional. If no container matching the type of certificate store you are adding exists, no containers will be available in the dropdown (see Certificate Store Container Operations on page 435). Leave blank if you do not wish the certificate store to be associated with a specific store container.

6. The remaining fields on the dialog will vary depending on the configuration of your custom certificate store type. Common fields include:

- **Client Machine**

  Typically the fully qualified domain name or IP address of the target server or device on which the certificate store is located.

  Important: In some cases, it's necessary to use the actual hostname of the target server in the **Client Machine** field rather than a DNS alias (either "A" or CNAME records). This is necessary when the orchestrator uses PowerShell remoting (WinRM) for some of the machine certificate store functions, which relies on Kerberos authentication. Kerberos authentication requires that the target machine has a service principal name (SPN) in the HTTP/ format assigned to the target's machine account. This will be present by default (as part of the HOST/ format record) as long as the HTTP/ format SPN has not been manually assigned elsewhere. Using an alias gets into complexities of setting up appropriate SPNs and assuring that there are not duplicate SPNs in the environment. If you wish to manage the target server hosting Keyfactor Command, you will need to use a DNS alias for either your Keyfactor Command server or the IIS store access. Contact Keyfactor for design assistance.

  Note: This field cannot be modified on an edit.

- **Store Path**

  The path to the certificate store, sometimes including the store file name, on the target server or device.

  For Java keystores, for example, this would be a file system path and file name (e.g. /opt/app/mystore.jks or C:\My App\mystore.jks). Paths and filenames entered for Linux/UNIX machines are case sensitive.

  For CA bundles on F5 devices, this is the bundle into which you want to install the certificate (e.g. /Common/myca-bundle). The Store Path name is case sensitive for some targets and devices, so, for example, if the partition name on the F5 is *Common* it must be entered in the Store Path field as *Common* rather than *common*.

  For IIS bound certificate stores, this is the name of the Windows certificate store (My = Local Computer *Personal* store, Web Hosting = Local Computer *Web Hosting* store).

  Check the details of the documentation for your custom-built extension to determine the data to enter here.

  Note: This field cannot be modified on an edit.

- **Orchestrator**

The name that the Keyfactor Universal Orchestrator used when registering with Keyfactor Command. The orchestrator must be approved in order to appear here (see Approving or Disapproving Orchestrators on page 500).

- **Server Username**

  The source from which to load a user valid on the target server or device with sufficient permissions to read the store location and open the file(s) and the name of that user, if applicable. In the Server Username dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. The *No Value* option is typically not supported for server authentication.

  For F5, this as a user with Administrator permissions. Although a user with Resource Administrator permissions is sufficient when using the F5 methods that use the SOAP API, the F5 methods that use the REST API require full Administrator permissions.

  For AWS using IAM authentication, this is the API access key for your web service.

  This field only appears if your custom certificate store type indicates that it *Needs Server*.

- **Server Password**

  The source from which to load a valid password for the user used to access the server and the password for that user, if applicable. In the Server Password dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. The *No Value* option is typically not supported for server authentication.

  This field only appears if your custom certificate store type indicates that it *Needs Server*.

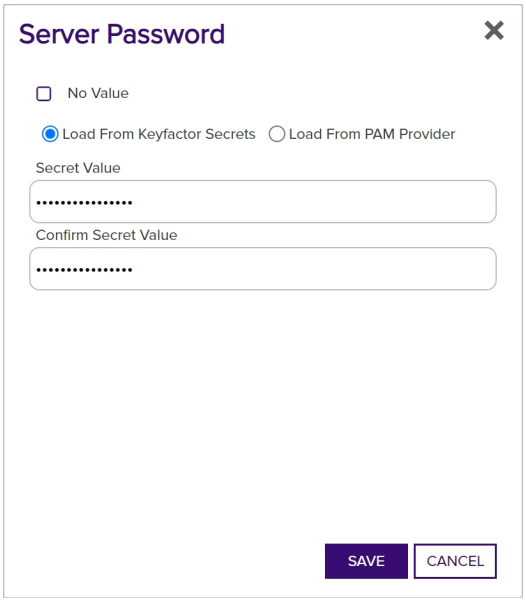


Figure 240: Set Password Dialog

A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends

using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

> ⚠ **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lower-case letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

CyberArk

Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

- Safe—The name of the safe the credential resides in.
- Object—The name of the username or password object in the safe.
- Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

- **Use SSL**

  Use SSL to communicate to the remote target or device.

  If the remote target is an F5 device and you're using the F5 extension, the device must trust the CA that issued the certificate used to protect the Keyfactor Command server or you must set the *Ignore Server SSL Warnings* application setting to *True* (see Application Settings: Agents Tab on page 614).

  If the remote target is a Windows server and you're using the Remote File or IIS extension, WinRM on the target must be configured to support HTTPS and have been configured with an SSL certificate (see Configure Windows Targets for Remote Management on page 2935).

  This field only appears if your custom certificate store type has been configured with the *Use SSL* custom field. Your field name may vary depending on the configuration of your custom certificate store type.

- **Create Certificate Store**

  If this box is checked, a new certificate store will be created on the target with the specified configuration. This option only appears if your custom certificate store type has been configured with Create for *Supported Job Types*.

- **Store Password.**

  The source from which to load a valid password for the certificate store itself rather than the server as a whole. In the Store Password dialog, the options are No Value, Load From Keyfactor Secrets, and Load From PAM Provider.

  Select **No Value** if your certificate store does not have a password configured.

  A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

  > ⚠ **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password

has at least 12 characters (more is better) and multiple character classes (lower-case letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

CyberArk

Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

- Safe—The name of the safe the credential resides in.
- Object—The name of the username or password object in the safe.
- Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

- **Type**

For Java keystores using the Keyfactor Java Agent, select the Type from the dropdown. The available types are:

- JKS

  Standard Java keystore.

- PKCS12

  PKCS12 type files (e.g. P12 or PFX), which are discoverable with the Keyfactor Java Agent using compatibility mode introduced in Java version 1.8.

- Windows-My

  Windows local machine personal certificate store. This option is only supported with a custom extension based on the AnyAgent framework. The Keyfactor Java Agent does not include functionality to manage this type of store.

The dialog may present additional fields depending on the configuration of the Custom Fields for the certificate store type. Some additional fields that you may find for common Keyfactor custom-built extensions include:

- For F5 certificates stores, in the **Primary Node** field, enter the fully qualified domain name of the F5 device that acts as the primary node in a highly available F5 implementation. If you're using a single F5 device, this will often be the same value you entered in the Client Machine field.

  Tip: Configuration of the primary node is necessary to allow management jobs that update certificates on the F5 device to wait until the primary node is available before making their update. Inventory jobs are carried out against any available node.

- For F5 certificates stores, in the **Primary Node Check Retry Wait Seconds** field, either accept the default value of 120 seconds or enter a new value. This value represents the number of seconds the orchestrator will wait after a pending management job cannot be completed because the primary node cannot be contacted before trying to contact the primary node again to retry the job.
- For F5 certificates stores, in the **Primary Node Check Retry Maximum** field, either accept the default value of 3 retry attempts or enter a new value. This value represents the number of times the orchestrator will retry a pending management job that is failing because the primary node cannot be contacted before declaring the job failed.
- For F5 certificates stores, in the **Version of F5** dropdown, select the version of F5 this server is running. The F5 REST API is supported on version 13 and up.

  Tip: Select v15 for version 15 and above.

- For IIS bound and Windows certificate stores, in the **WinRm Protocol** field, select http to communicate between the orchestrator and the target over a non-secured WinRM

channel or https to communicate over a secured channel WinRM channel. Using HTTPS for WinRM requires that the target has been configured to support HTTPS and has been configured with an SSL certificate for WinRM. For example:

https://learn.microsoft.com/en-us/troubleshoot/windows-client/system-management-components/configure-winrm-for-https

- For IIS bound and Windows certificate stores, in the **WinRm Port** field, accept the default of port 5986 for HTTPS, enter port 5985 for the default HTTP WinRM port, or enter an alternate port if your WinRM configuration is using an alternate port.
- For IIS bound and Windows certificate stores, in the **SPN with Port field**, accept the default of false unless you're aware that the -IncludePortwithSPN switch is required for remote PowerShell connections in your environment.
- For Java keystores, PEM stores, and PKCS12 stores on Linux targets when creating a new certificate store (see *Create Certificate Store*) using the Remote File extension, in the **Linux File Permissions on Store Creation** field, set the file permissions that should be granted to the new certificate store given in numeric format (e.g. 644). The default is 600.
- For Java keystores, PEM stores, and PKCS12 stores on Linux targets when creating a new certificate store (see *Create Certificate Store*) using the Remote File extension, in the **Linux File Owner on Store Creation** field, set the file ownership that should be granted to the new certificate store given as a Linux username. The default is the user configured to authenticate to the server (see *Server Username*).
- For PEM stores, in the **Separate Private Key File Location** field, enter the full path to the private key on the machine, including the file name. Paths and filenames entered for Linux/UNIX machines are case sensitive.
- For PEM stores using the Remote File extension, in the **Trust Store** field, a value of True indicates the target store is a trust store and as such should only contain public keys. If a store is marked as a trust store and a job is run that attempts to add a private key for a certificate to the store, an error will occur indicating a problem. A value of False indicates the store can contain a single certificate with private key and chain. The default is False.
- For PEM stores using the Remote File extension, in the **Store Includes Chain** field, a value of True causes the full chain for the certificate to be delivered to the store. A value of False indicates that only the certificate and private key should be delivered to the store. The certificates are always placed in the store in the order of : End Entity, Issuing CA, Root CA. The default is False. The field is ignored if *Trust Store* is True.
- For PEM stores using the Remote File extension, in the **Is RSA Private Key** field, a value of True indicates the private key for the certificate store is a PKCS#1 RSA formatted key (it will be headed with a line that reads "BEGIN RSA PRIVATE KEY"). A value of False indicates the private key for the certificate store is either an encrypted or non-encrypted PKCS#8 formatted key (it will be headed with a line that reads either "BEGIN PRIVATE KEY" or "BEGIN ENCRYPTED PRIVATE KEY"). If set to True, the *Store*

Password must be set to No Value. The default is False. The field is ignored if *Trust Store* is True.

7. In the Inventory Schedule fields, select an inventory schedule for the store, if desired. You can choose to run the inventory *Daily*, on an *Interval*, *Immediately*, *Exactly Once*, or set inventorying to *Off*.
    - If you select **Daily**, you can set the time of day when the inventory should begin every day.
    - If you select **Interval**, you can select a scan frequency of anywhere from every 1 minute to every 12 hours.
    - If you select **Immediate**, the inventory will run within a few minutes of saving the record and will run only once. After this, the inventory schedule will be cleared.
    - If you select **Exactly Once**, you can select a date and time at which to run the inventory job. After the job has run, the inventory schedule will be cleared.
    - Select **Off** to disable the inventory job.

    If you are using Certificate Store Containers (see Certificate Store Containers on page 432) to manage your stores and their schedules you do not need to set an inventory schedule here.
8. Click **Save** to save the new or edited certificate store location.

Deleting a Certificate Store

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Read
Certificate Stores > Modify

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

To delete a certificate store:

1. In the Management Portal, browse to *Locations* > *Certificate Stores*.
2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).
3. On the Certificate Stores tab, highlight the row(s) in the certificate store grid of the store(s) to delete and click **Delete** at the top of the grid or right-click the store location in the grid and choose **Delete** from the right-click menu. The right-click menu supports operations on only one store at a time.
4. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Note: This doesn't delete the actual certificate store on the target server, just the Keyfactor Command definition of it.

Viewing a Certificate Store

Users without modify permissions to certificate stores will see a *View* option instead of an *Edit* option on the Certificate Stores page to allow them to see a read-only view of the certificate store configuration details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Agents > Management > Read
Certificate Stores > Read

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

To view the details of a certificate store:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).
3. On the Certificate Stores tab, highlight the row in the certificate store grid of the store for which to view certificate store details and click **View** at the top of the grid or right-click the store location in the grid and choose **View** from the right-click menu.

The fields are the same as those described for adding or editing a certificate store (see Adding or Modifying a Certificate Store on page 413), but none of the fields are editable when using the *View* option.

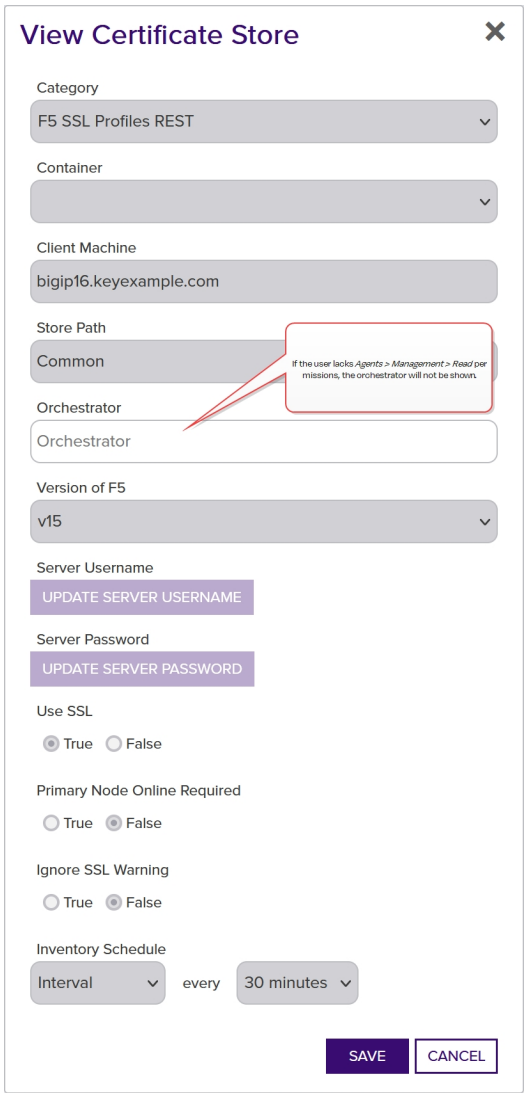


Figure 241: View Details for a Certificate Store

Certificate Store Reenrollment

The Reenrollment option is available for select built-in and custom-built extensions provided by Keyfactor on GitHub or can be included in your fully custom extension. Examples of built-in and Keyfactor custom-built extensions that support reenrollment include:

- Akamai Certificate Provisioning System (CPS) on GitHub for use with the Keyfactor Universal Orchestrator
- IIS Certificate Store Manager on GitHub for use with the Keyfactor Universal Orchestrator
- PEM and Java certificate stores managed by the Keyfactor Java Agent.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificates > Enrollment > CSR
Certificate Stores > Read
Certificate Stores > Modify

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

In addition, either the user scheduling the reenrollment job or the user configured to provide authentication to the CA (see Authorization Methods Tab on page 367) must have enrollment permissions configured on the CA and template.

To begin a reenrollment:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).
3. On the Certificate Stores tab, highlight the row in the certificate store grid of the store to reenroll and click **Reenrollment** at the top of the grid or right-click the store location in the grid and choose **Reenrollment** from the right-click menu.
4. On the Reenrollment dialog, enter a **Subject Name** for the new certificate using X.500 format and populate any additional fields as required by the certificate store type (on the *Entry Parameters* tab; see Adding or Editing a Certificate Store Type on page 701).
5. If desired, select a **Certificate Authority** to direct the enrollment request to and/or **Template** for the request. In the **Template** dropdown, only templates that are available for enrollment from the selected certificate authority will appear.

   Note: If you don't select a template or CA for reenrollment, the values configured for the *Template For Submitted CSRs* and/or *Certificate Authority For Submitted CSRs* application setting(s) (see Application Settings: Agents Tab on page 614) will be used.

   Tip: If an expected template does not appear in the **Template** dropdown, confirm that it is available for enrollment from the selected CA, has been configured for CSR enrollment in Keyfactor Command, and has appropriate enrollment permissions at the Keyfactor Command, CA and template level for the user making the reenrollment request or relevant service account depending on the authorization method for enrollment (see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832 in the *Keyfactor Command Server Installation Guide*).
6. Click Done to submit the request.

The reenrollment job will be scheduled to run immediately. Visit the Orchestrator Jobs page to check on the progress of the job (see Orchestrator Job Status on page 510).

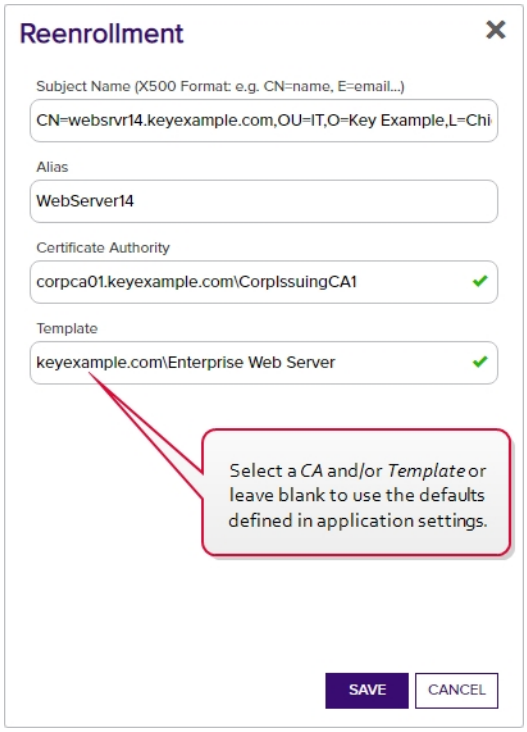


Figure 242: Enter a Information for Reenrollment

Setting a New Password on a Certificate Store

The option to reset the password on a certificate store updates the data for the certificate store as stored in the Keyfactor Command database but does not make any modifications to the certificate store itself. This option is available from the right-click menu only.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Read
Certificate Stores > Modify

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

To reset the password for a certificate store:

1. In the Management Portal, browse to *Locations* > *Certificate Stores*.
2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).

3. On the Certificate Stores tab, highlight the row in the certificate store grid of the store to update and choose **Set New Password** from the right-click menu.

4. Enter and confirm the new password and click **Save**.

Assigning a Certificate Store to a Container

Before assigning a certificate store to a container, you need to create the container (see Certificate Store Containers on page 432). If you select multiple certificate stores to assign to a container at once, they must all be stores of the same type (e.g. PEM).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Read
Certificate Stores > Modify

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

To assign a certificate store to a container:

1. In the Management Portal, browse to *Locations > Certificate Stores*.

2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).

3. On the Certificate Stores tab, highlight the row(s) in the certificate store grid of the store(s) to be assigned to the container and click **Assign Container** at the top of the grid or right-click the store location in the grid and choose **Assign Container** from the right-click menu. The right-click menu supports operations on only one store at a time.

4. Select a certificate store container in the Container Name field and click **Save**.

Viewing Inventory for a Certificate Store

Once at least one inventory job has been completed for a given certificate store, you can view the certificates imported from the store.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
PAM > Read
Certificate Stores > Read
Certificate > Collections > Read

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

To view the inventoried certificates for a store:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).
3. On the Certificate Stores tab, highlight the row in the certificate store grid of the store for which to view inventory and click **View Inventory** at the top of the grid or right-click the store location in the grid and choose **View Inventory** from the right-click menu.

On the left of the inventory viewing dialog you can select a certificate from the store to view. On the right of the dialog you can see details about that certificate, including the metadata associated with the certificate. In the Certificate Selection area of the screen, you can select between the chain certificates for the selected certificate and the end entity certificate, for certificates stored with a chain.

At the bottom of the inventory viewing dialog, click **Query Certificate**, after selecting the certificate to view, to open a new browser tab to the certificate search page with the search criteria specifying only the certificate you selected. The *Revoked* and *Expired* check boxes will automatically be included if either condition is true for that certificate.

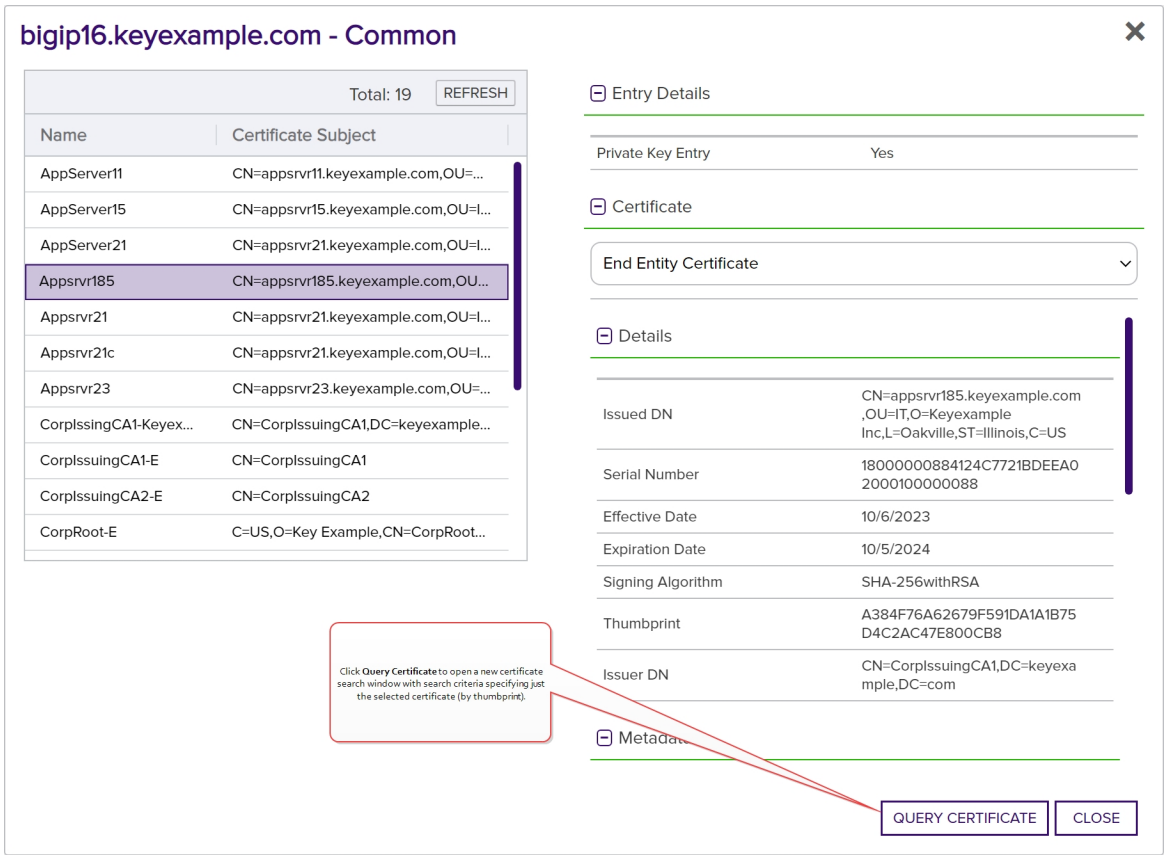


Figure 243: View Inventoried Certificates for a Certificate Store

Scheduling Inventory for a Certificate Store

Scheduling inventory for a certificate store allows Keyfactor Command to inspect the certificates inside a given store and add them to the Keyfactor Command database.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Read
Certificate Stores > Schedule

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

To schedule inventory:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).

3. On the Certificate Stores tab, highlight the row(s) in the certificate store grid of the store(s) for which you want to schedule inventory and click **Schedule Inventory** at the top of the grid, or choose **Schedule Inventory** from the right-click menu. The right-click menu supports operations on only one store at a time.

4. In the Certificate Store Inventory Schedule dialog, select a schedule for the store(s). You can choose to run the inventory *Daily*, on an *Interval*, *Immediately*, *Exactly Once*, or set inventorying to *Off*.
    - If you select **Daily**, you can set the time of day when the inventory should begin every day.
    - If you select **Interval**, you can select a scan frequency of anywhere from every 1 minute to every 12 hours.
    - If you select **Immediate**, the inventory will run within a few minutes of saving the record and will run only once. After this, the inventory schedule will be cleared.
    - If you select **Exactly Once**, you can select a date and time at which to run the inventory job. After the job has run, the inventory schedule will be cleared.
    - Select **Off** to disable the inventory job.

You have the option to not schedule inventory on a store-by-store basis and instead create containers and set inventory schedules that will apply to all the stores added to each container. See Certificate Store Containers on the next page for information on creating containers.

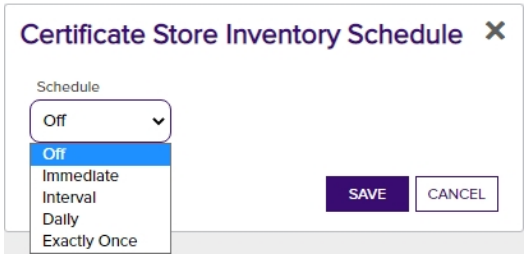


Figure 244: Schedule Inventory for a Certificate Store Location

Querying Certificates for a Certificate Store

The **Query Certificate Store** option allows you to open a new browser tab with an instance of the Management Portal open to the certificate search page pre-populated with search criteria to locate the certificates found in that certificate store.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Modify
Certificate Stores > Read

Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. You can use a mixture with, for example, global certificate permissions and container-level certificate store permissions. See

Certificate Collection Permissions on page 627 and Container Permissions on page 629 for more information about global vs collection and container permissions.

To use the query certificate store option:

1. In the Management Portal, browse to *Locations* > *Certificate Stores*.
2. On the Certificate Stores page, select the Certificate Stores tab (the default when you first visit the page).
3. On the Certificate Stores tab, highlight the row in the certificate store grid of the store for which you want to query certificates and click **Query Certificate Store** at the top of the grid, or choose **Query Certificate Store** from the right-click menu.

Certificate Store Containers

Certificate store containers allow you to collect similar stores together to provide organization, allow for simplified bulk operations and control access.

Using the Containers Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

Name

Complete or partial matches with the name of the certificate store container.

Schedule

Whether the certificate store container has a schedule defined, true/false.

CertStoreType

The certificate store type of the container.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

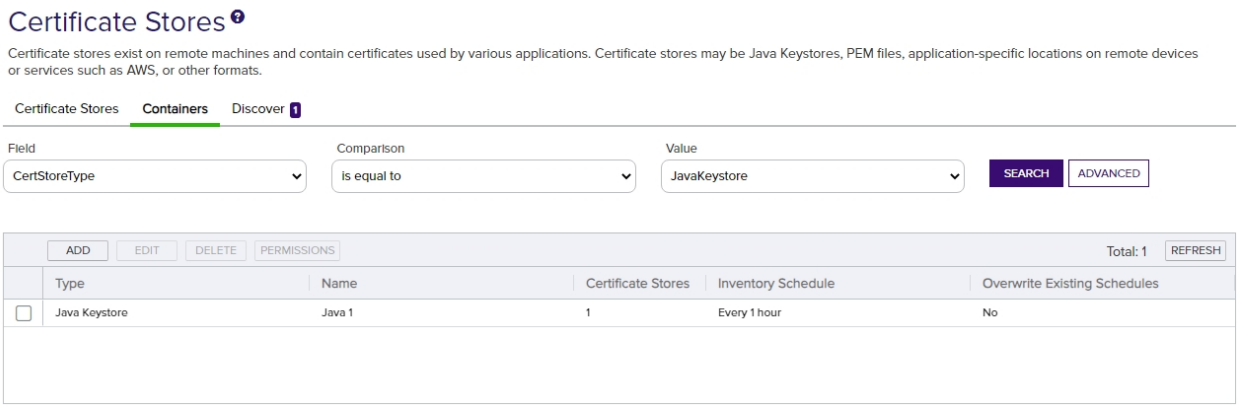


Figure 245: Certificate Store Container Search

The search results can be sorted by clicking on a column header in the results grid for most columns. Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may be adjusted by click-holding and dragging the line separating two column headers.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

Certificate Store Container Operations

Certificate store container operations include creating or editing containers—including scheduling inventory for the container—and deleting containers.

Adding or Modifying a Certificate Store Container

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Modify
Certificate Stores > Read

To add or edit a certificate store container:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Stores page, select the Containers tab.

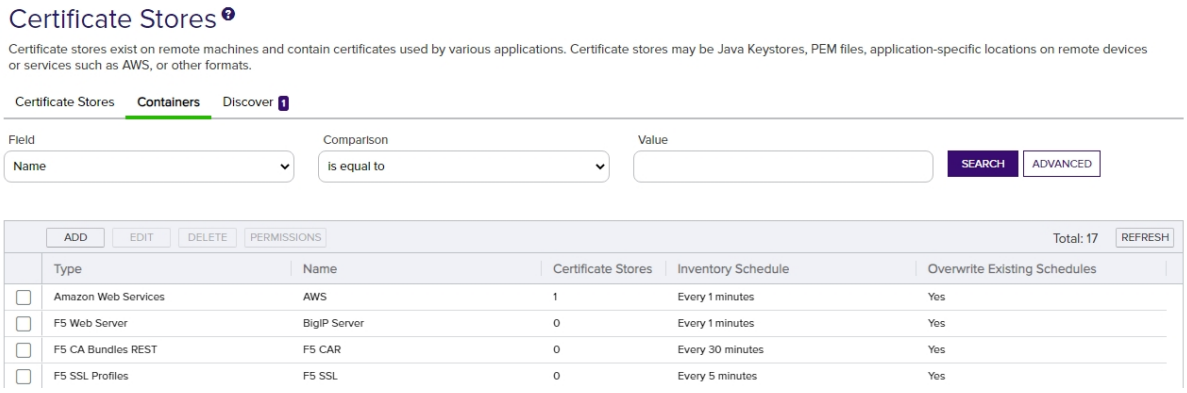


Figure 246: Certificate Store Containers

3. On the Containers tab, click **Add** to create a new container, or click **Edit** from either the top or right-click menu to modify an existing one.
4. In the Schedule Container dialog, select the appropriate **Type** for the container from the drop-down. This field cannot be modified on an edit.

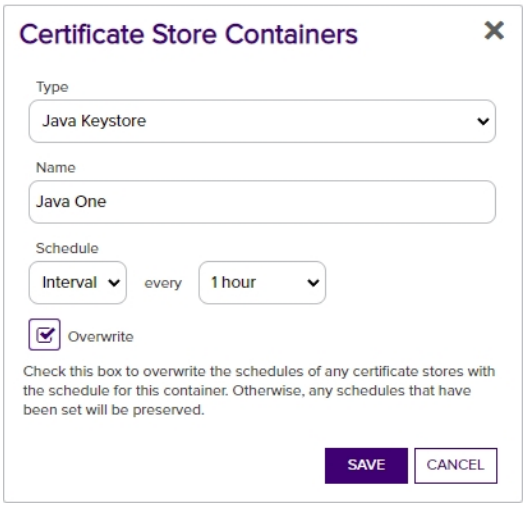


Figure 247: Define a Certificate Store Container

5. Enter a name for the container in the **Name** field.
6. In the **Inventory Schedule** fields, select an inventory frequency to apply as a default to certificate stores added to the container. The choices are:
    - Daily at a selected time
    - At intervals of anywhere from every one minute to every 12 hours
    - Off
7. If desired, check the **Overwrite Existing Schedules** box. This option will apply the schedule from the container to any stores in the container, including those that already have a schedule, whenever the container schedule is updated.
8. Click **Save** to save the container.

Deleting a Container

Deleting a container that contains certificate stores does not delete the associated certificate stores. The certificate stores will remain and be disassociated from the container.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Modify
Certificate Stores > Read

To delete a certificate store container:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Stores page, select the Containers tab.

3. On the Containers tab, highlight the row in the certificate store containers grid of the container to delete and click **Delete** at the top of the grid or right-click the container in the grid and choose **Delete** from the right-click menu. Only one container may be deleted at a time.

4. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Container Permissions

Although permissions for a container can be viewed or modified using the permission option on the certificate store containers tab, Keyfactor recommends best practice is to manage permissions as part of the overall permission configuration on the Security Roles and Claims on page 622 page, as additional system-wide configuration settings are available there that cannot be viewed or modified from this page. For more information, see Container Permissions on page 629 and Security Roles and Claims on page 622.

Certificate Store Discovery

The certificate store discovery feature is used to scan machines and devices for existing certificates and certificate stores, which can then be configured for management in Keyfactor Command. Certificate store discovery is supported for the following built-in certificate stores and Keyfactor-provided custom-built extensions on GitHub:

- PEM and Java certificate stores discovered by the Keyfactor Java Agent. Only stores on which the service account running the Keyfactor Java Agent has at least read permissions will be returned on a discover job.
- F5 CA bundles and F5 SSL certificates discovered using the F5 Certificate Store Manager extension from Keyfactor GitHub and the Keyfactor Universal Orchestrator.
- Java Keystores, PKCS12 files, PEM files, DER files, and IBM Key Database files discovered using the Remote File Certificate Store Management extension from Keyfactor GitHub and the Keyfactor Universal Orchestrator.

The small number that appears on the tab to the right of the word Discover indicates how many discovered stores there are, if any. This acts as a reminder to check the discover tab for stores after a discovery job is complete.

Scheduling a Certificate Store Discovery Job

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Modify
Certificate Stores > Read
Certificate Stores > Schedule
PAM > Read

To use the certificate store discovery feature:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Store page, select the Discover tab.
3. On the Discover tab, click **Schedule**.
4. In the Schedule Discovery dialog, select the type of certificate store job in the **Category** drop-down. The values that appear here are the display names for the built-in certificate store types and any custom certificate store types you entered to match your custom extensions.

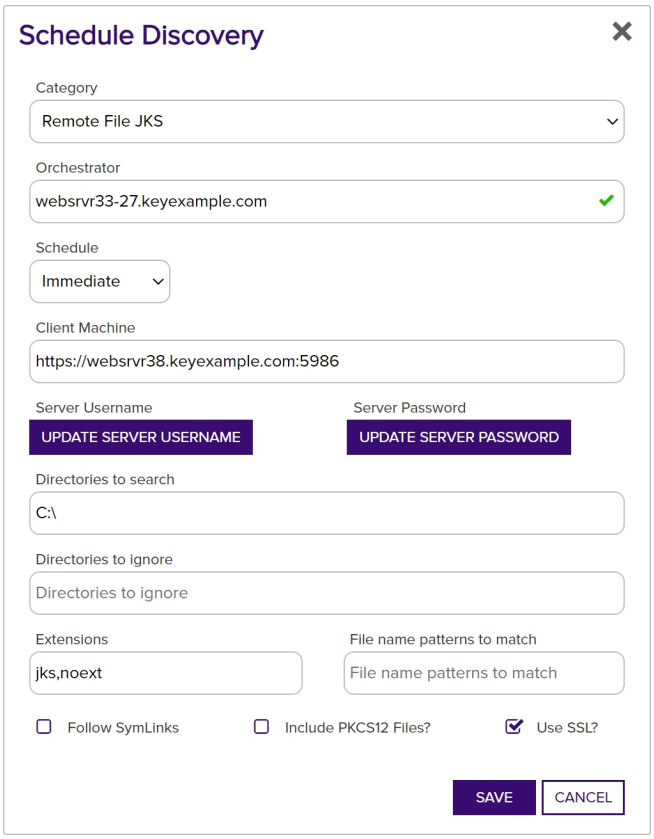


Figure 248: Schedule Java Keystore Discover Job for Remote File Extension

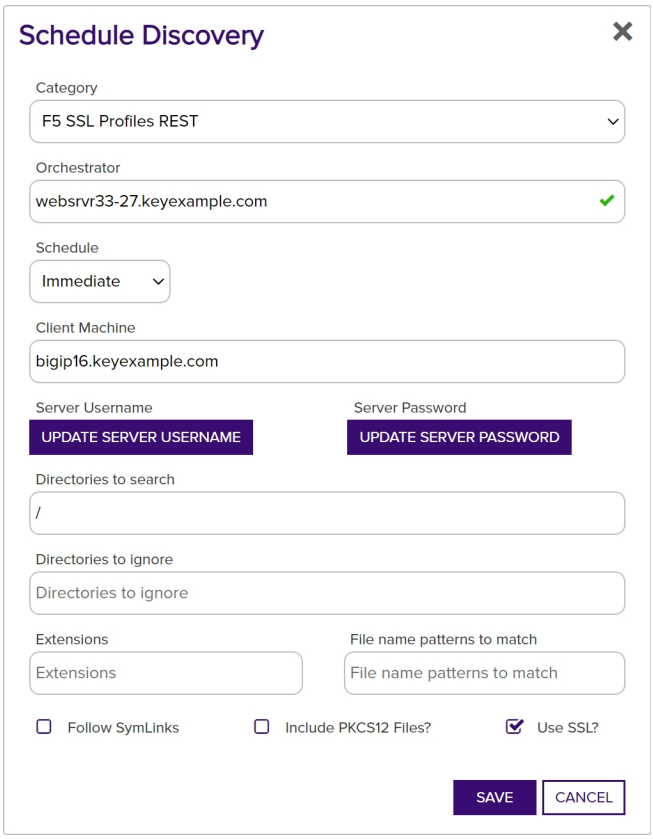


Figure 249: Schedule F5 SSL Discover Job for F5 Extension

5. In the **Orchestrator** field, select the fully qualified domain name of the approved Keyfactor Universal Orchestrator managing the scanning. This field is required.

6. In the **Schedule** dropdown, select either *Immediate*, to run the discover job within a few minutes of saving it, or *Exactly Once*, to select a date and time for the job. The default is Immediate.

7. In the **Client Machine** field, enter the fully qualified domain name or IP address of the target server or device to be scanned.

8. Click **Set Server Username** and, in the Server Username dialog, choose the source from which to load a user valid on the target server or device with sufficient permissions to read the store locations and open files (for F5 this should be Administrator permissions). In the Server Username dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. The *No Value* option is typically not supported for remote file targets and F5 devices.

   Note: Although a user with Resource Administrator permissions is sufficient when using the F5 methods that use the SOAP API, the F5 methods that use the REST API require full Administrator permissions.

A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

CyberArk

Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

- Safe—The name of the safe the credential resides in.
- Object—The name of the username or password object in the safe.
- Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

9. Click **Set Server Password** and, in the Server Password dialog, choose the source from which to load the password for the user specified with Set Server Username. In the Server Password dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. The *No Value* option is typically not supported for remote file targets and F5 devices.

   A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

   > ⚠ **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

   Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

   Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

   ### CyberArk

   Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

   - Safe—The name of the safe the credential resides in.
   - Object—The name of the username or password object in the safe.
   - Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

   ### Delinea

   Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

10. In the **Directories to search** field, specify the directory or directories to search. Multiple directories should be separated by commas. The data to enter here will vary depending on the certificate store(s) being discovered. For Keyfactor GitHub extensions, check the extension documentation for more information. This field is required.

    In general:

    - Java, PEM, PKCS12, DER, IMB

      Enter at a minimum either "/" for a Linux server or "c:\" for a Windows server (without the quotation marks).
    - F5

      Enter "/" (without the quotation marks). (This field is required but ignored by the discovery job for F5, so any value entered here will do.)

11. Populate the remaining optional fields as needed. See Table 14: Discovery Options.
12. Click **Save** to schedule the discovery task. Once the scan begins, it may take several minutes to complete.
13. Return to the Discover tab for the results of the scan. Check the Orchestrator Jobs page (see Orchestrator Job Status on page 510) to review jobs in progress.

Managing Discovered Certificate Stores

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Modify
Certificate Stores > Read
PAM > Read

To manage discovered certificate stores:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Stores page, select the Discover tab.

3. On the Discover tab, highlight one or more store row(s) in the grid and click **Manage** at the top of the grid or right-click the store in the grid and choose **Manage** from the right-click menu. Discovered certificate stores that require entry of either a server username and password or store password (or PAM credential access information) during the approval process must all share the same password or PAM information if you select more than one for approval at the same time. The right-click menu supports operations on only one store at a time.

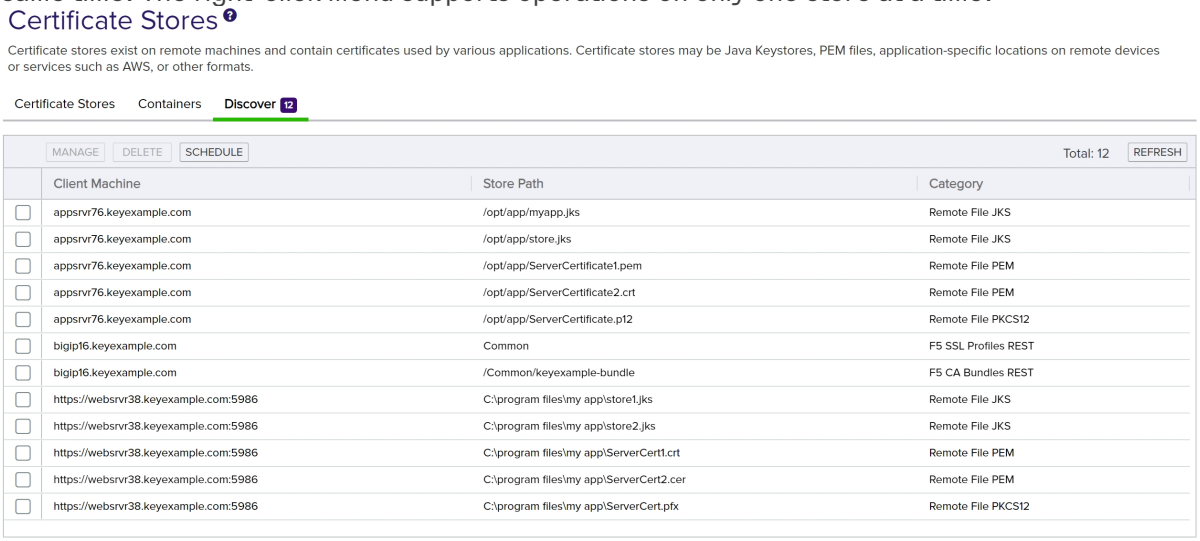


Figure 250: Discovered Certificate Stores

4. The fields that appear on the Manage Certificate Stores dialog will vary depending on the certificate store type. If you're using a Keyfactor custom-built extension from GitHub, be sure to consult the documentation on GitHub for the specific extension. The following are some examples.

Java Keystore with the Remote File Extension

- If desired, select a **Container** from the dropdown.
- Click the **Set Password** button to enter the password for the keystore. In the Password dialog, the options are No Value, Load From Keyfactor Secrets, and Load From PAM Provider.

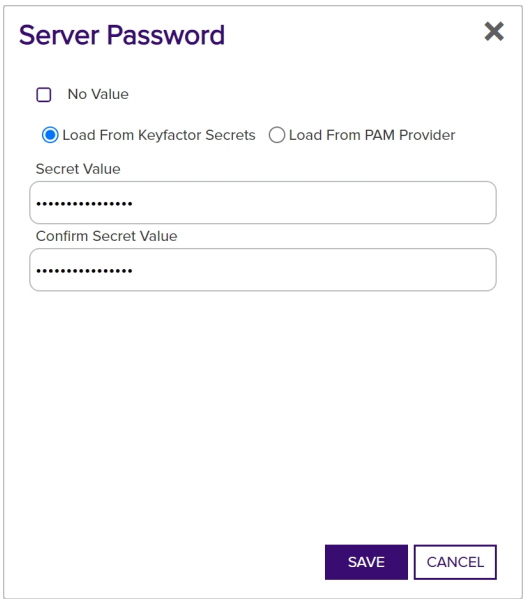


Figure 251: Java Keystore Set Password

Select **No Value** if your keystore does not have a password configured.

A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lower-case letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

CyberArk

Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

- Safe—The name of the safe the credential resides in.
- Object—The name of the username or password object in the safe.
- Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

- The **Linux File Permissions on Store Creation** and **Linux File Owner on Store Creation** fields apply to the creation of new certificate stores on the target, and do not apply to discovered stores. These fields may be left blank.
- Click **Set Server Username** and, in the Server Username dialog, choose the source from which to load a user valid on the target server or device with sufficient permissions to read the store locations and open files (for F5 this should be Administrator permissions). In the Server Username dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. The *No Value* option is typically not supported for remote file targets.

Note: Although a user with Resource Administrator permissions is sufficient when using the F5 methods that use the SOAP API, the F5 methods that use the REST API require full Administrator permissions.

- Click **Set Server Password** and, in the Server Password dialog, choose the source from which to load the password for the user specified with Set Server Username. In the Server Password dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. The *No Value* option is typically not supported for remote file targets.

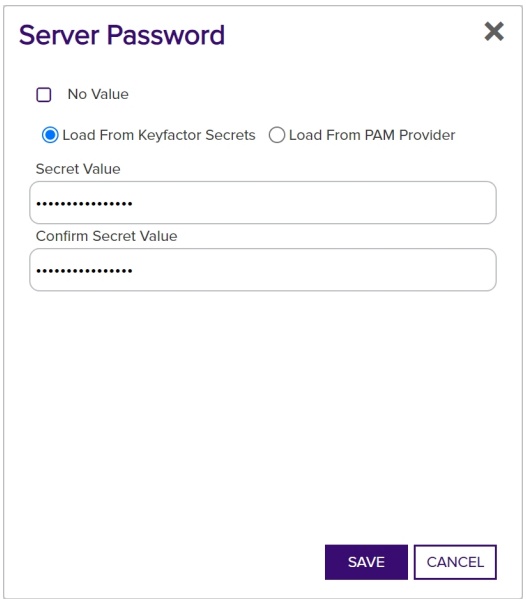


Figure 252: F5 SSL Profiles Set Password

A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lower-case letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access

Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

CyberArk

Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

- Safe—The name of the safe the credential resides in.
- Object—The name of the username or password object in the safe.
- Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

- In the **Use SSL** section, select *True* to use SSL to communicate with the remote target, if desired. For more information, see Table 14: Discovery Options.

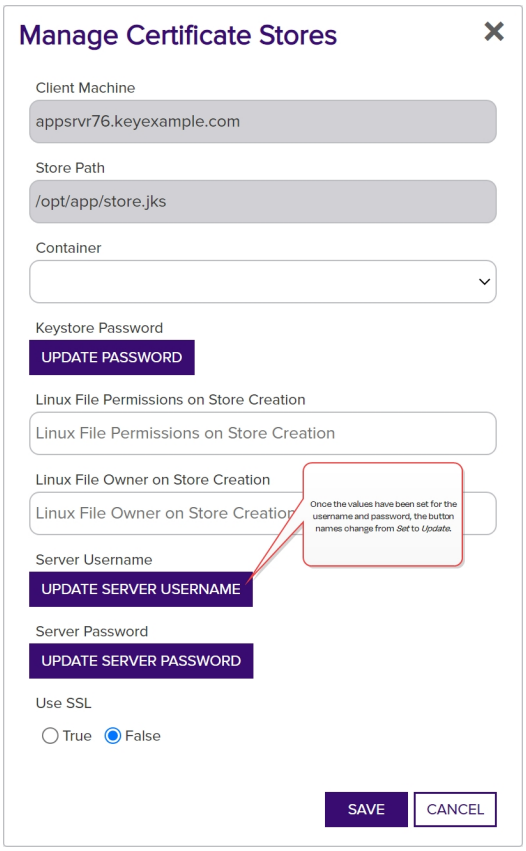


Figure 253: Manage a Discovered Java Certificate Store

F5 SSL Profile Certificate with the F5 Extension

- If desired, select a **Container** from the dropdown.
- In the **Primary Node** field, enter the fully qualified domain name of the F5 device that acts as the primary node in a highly available F5 implementation. If you're using a single F5 device, this will often be the same value you entered in the Client Machine field.

  Tip: Configuration of the primary node is necessary to allow management jobs that update certificates on the F5 device to wait until the primary node is available before making their update. Inventory jobs are carried out against any available node.

- In the **Primary Node Check Retry Wait Seconds** field, either accept the default value of 120 seconds or enter a new value. This value represents the number of seconds the orchestrator will wait after a pending management job cannot be completed because the primary node cannot be contacted before trying to contact the primary node again to retry the job.

- In the **Primary Node Check Retry Maximum** field, either accept the default value of 3 retry attempts or enter a new value. This value represents the number of times the orchestrator will retry a pending management job that is failing because the primary node cannot be contacted before declaring the job failed.
- In the **Version of F5** dropdown, select the version of F5 this server is running. The F5 REST API is supported on version 13 and up.

  Tip: Select v15 for version 15 and above.

- Click **Set Server Username** to choose the source from which to load a user valid on the F5 device with *Administrator* permissions. In the Server Username dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. The *No Value* option is typically not supported for F5 stores.

  Note: Although a user with Resource Administrator permissions is sufficient when using the F5 methods that use the SOAP API, the F5 methods that use the REST API require full Administrator permissions.

- Click **Set Server Password** to choose the source to load a valid password for the server. In the Server Password dialog, the options are Load From Keyfactor Secrets or Load From PAM Provider. The *No Value* option is typically not supported for F5 stores.

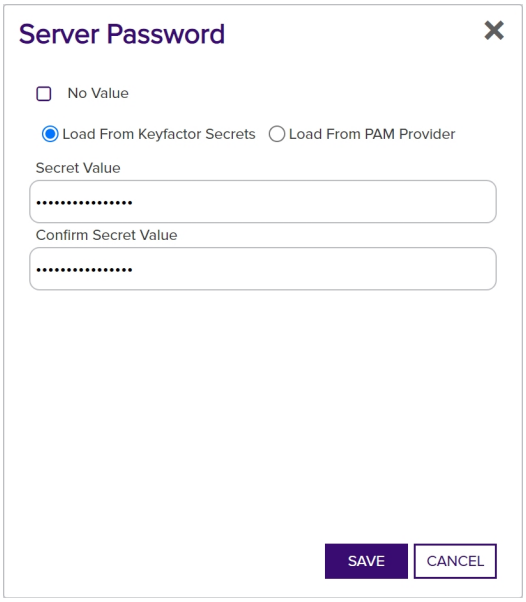


Figure 254: F5 SSL Profiles Set Password

A Keyfactor secret is a user-defined password or other information that is encrypted and stored securely in the Keyfactor Command database. Although Keyfactor recommends using Privileged Access Management (see Privileged Access Management (PAM) on page 742) as a more secure solution to secure information, Keyfactor Secret is an option for customers that do not already have a relationship with a PAM provider such as CyberArk or Delinea.

> **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lower-case letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Select the **Load From Keyfactor Secrets** radio button as the *Secret Source* if you want Keyfactor Command to encrypt and store the password in the Keyfactor Command database. Enter and confirm a password.

Select the **Load from PAM Provider** radio button as the *Secret Source* if you want to store the password in a supported third-party PAM solution (see Privileged Access Management (PAM) on page 742). The remaining fields on the dialog will vary depending on the PAM provider. For example:

CyberArk

Select your CyberArk provider in the **Providers** dropdown if your PAM provider is CyberArk (see Adding or Modifying a PAM Provider on page 750). The remaining fields in the dialog will then be:

- Safe—The name of the safe the credential resides in.
- Object—The name of the username or password object in the safe.
- Folder—The path and name of the folder that stores the object (e.g. Root or Root\MyDir).

Delinea

Select your Delinea provider in the **Providers** dropdown if your PAM provider is Delinea (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- Secret Server Secret ID—The numeric ID of the secret to retrieve from Secret Server.
- Secret Field Name—The name of the field to use when retrieving a secret form Secret Server.

Hashicorp

Select your Hashicorp Vault provider in the **Providers** dropdown if your PAM provider is Hashicorp Vault (see Adding or Modifying a PAM Provider on page 750). The remaining field in the dialog will then be:

- KV Secret Key—The key of the secret to retrieve from the Hashicorp Vault.
- KV Secret Name—The name of the secret to retrieve from the Hashicorp Vault.

- In the **Use SSL** section, select *True* to use SSL to communicate with the F5 device or cluster, if desired. For more information, see Table 14: Discovery Options.

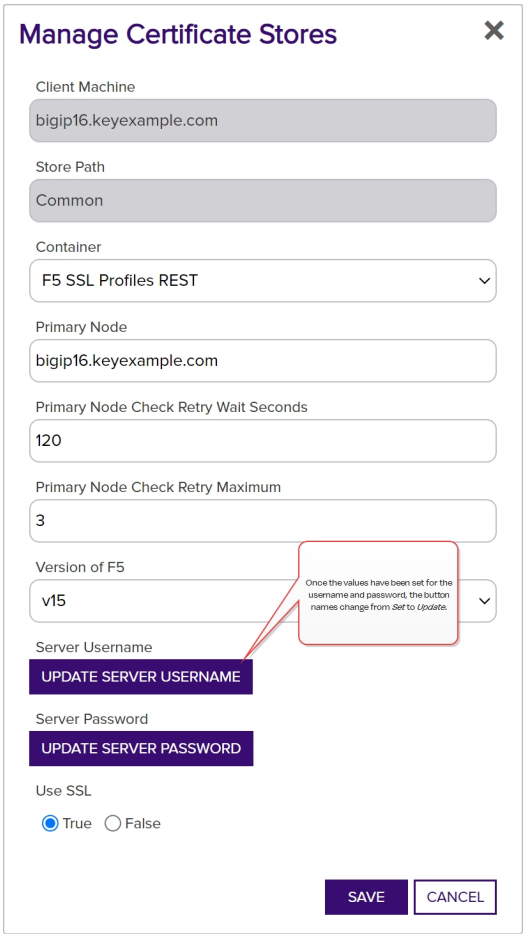


Figure 255: Manage a Discovered F5 SSL Profile Certificate

Deleting a Discovered Certificate Store

Discovered certificate stores can be deleted one at a time or in multiples.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

Certificate Stores > Modify
Certificate Stores > Read

To delete a discovered certificate store:

1. In the Management Portal, browse to *Locations > Certificate Stores*.
2. On the Certificate Stores page, select the Discover tab.
3. On the Discover tab, highlight the row(s) in the discover grid of the store(s) to delete and click **Delete** at the top of the grid or right-click the store location in the grid and choose **Delete** from the right-click menu. The right-click menu supports operations on only one store at a time.
4. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

The following table includes only default fields for discovery jobs, not any custom fields specific to a certain certificate store type.

Table 14: Discovery Options

Option	Description
Category	Select the type of certificate store to scan.
Orchestrator	Select the fully qualified domain name of the Keyfactor Universal Orchestrator managing the scanning. This field is required.
Schedule	Specify the schedule for the scan—Immediate or Exactly Once. If you select Exactly Once, select a date and time for the scan. The default is Immediate.
Client Machine	Specify the fully qualified domain name or IP address of the remote target, device or cluster to be scanned for certificates. This field is required.
Server Username	Set the username used to authenticate to the remote target, device or cluster.
Server Password	Set the password used to authenticate to the remote target, device or cluster.
Directories to search	Specify the directory or directories to be searched. Multiple directories should be separated by commas. All directories specified to which the specified user (see *Server Username*) has read rights will be searched other than the excluded directories specified using the *Directories to ignore* option. It is not necessary to use quotation marks around directory paths containing spaces. For F5, the path should be specified as "/" (without the quotation marks). This field is required.
Directories to ignore	Specify any directories that should not be included in the search. Multiple directories should be separated by commas. It is not necessary to use quotation marks around directory paths containing spaces.

Option	Description
Extensions	Specify file extensions for which to search. For example, search for files with the extension *jks* but not *txt*. The dot should not be included when specifying extensions. To include files without extensions, include *noext* in the extension list. For example: `jks, noext`
File name patterns to match	Specify all or part of a string against which to compare the file names of certificate store files and return only those that contain the specified string. It is not necessary to use quotation marks around strings containing spaces.
Follow SymLinks	If this option is specified, the tool will follow symbolic links on Linux and UNIX operating systems and report both the actual location of a found certificate store file in addition to the symbolic link pointing to the file. This option is ignored for searches of Windows-based targets.
Include PKCS12 Files	If this option is specified, the tool will use the compatibility mode introduced in Java version 1.8 to locate both JKS and PKCS12 type files if supported by the extension. This option applies only to Java keystore discover jobs.
Use SSL	Use SSL to communicate to the remote target, device or cluster. If the remote target is an F5 device and you're using the F5 extension, the device must trust the CA that issued the certificate used to protect the Keyfactor Command server or you must set the *Ignore Server SSL Warnings* application setting to *True* (see Application Settings on page 601). If the remote target is a Windows server and you're using the Remote File or IIS extension, WinRM on the target must be configured to support HTTPS and have been configured with an SSL certificate (see Configure Windows Targets for Remote Management on page 2935).

2.1.8.4 SSL Discovery

SSL network discovery and monitoring is used to survey designated internet-facing or internal IP addresses and ports to locate and import certificates, as well as alert certificate owners when the certificates are nearing expiration or are not found. Discovery jobs scan network segments to locate certificates at TLS endpoints; whereas, monitoring jobs inspect certificates for health and expiration and notify recipients regarding the status of the certificates. With the introduction of the Keyfactor Universal Orchestrator, SSL discovery can scan TLS 1.3 endpoints using any of the 5 ciphersuites referenced in appendix B.4 of RFC 8446.

SSL network discovery and monitoring scanning is performed by orchestrators that are assigned to orchestrator pools. An orchestrator pool contains orchestrators that support SSL discovery and monitoring capabilities for its networks. Orchestrator architecture allows for a pool of orchestrators to work in parallel to execute scan jobs. Based on defined schedules, Keyfactor Command creates discovery or monitoring scan jobs. Several scan jobs may be created from one large request. Orchestrators poll the Keyfactor Command Service to determine if scan jobs are available. Scan

jobs are then executed by available orchestrators. Keyfactor Command automatically distributes the scanning load across the orchestrators in the pool by generating and managing individual scan jobs. Additionally, the orchestrator that discovers the certificate can be different than the orchestrator that monitors the certificate.

The orchestrator SSL scanning process will attempt to scan with and without server name indication (SNI) for endpoints specified by host name during discovery scans and only use SNI during a monitoring scan if the endpoint has an SNI name from the discovery scan. Whenever an endpoint is defined to scan by its host name, the orchestrator will try to scan that endpoint twice, one normal scan against the endpoint and one using the supplied host name as the SNI extension.

Keyfactor Command is installed with a *Default Orchestrator Pool* that holds all the orchestrators that have been configured for SSL network discovery and monitoring. Custom orchestrator pools can be created as needed.

Note: The orchestrators in the network's orchestrator pool must have access to the network the pool is assigned to scan. Ideally, orchestrators are placed in close network proximity to the addresses they are configured to scan. Scanning across WAN or slow network links can impact performance and potentially miss certificates due to timeouts or network congestion. Additionally, firewalls between the orchestrators and their target networks need to be configured to allow connections to the scanned addresses and ports.

SSL network discovery and monitoring is divided into three areas:

- **Network Definitions**

  Network definitions are used to define a collection of networks that will be scanned by the designated orchestrator pool. Networks are defined using IP addresses, ports and hostnames. Within this option, you can schedule discovery and/or monitoring tasks. You can also configure networks to automatically tag a discovered endpoint with a certificate for monitoring.

- **Orchestrator Pools Definition**

  On the orchestrator pools definition tab you define a group of available orchestrators that support the SSL discovery and monitoring capabilities. For each orchestrator added to the orchestrator pool, you can select discover and/or monitor option(s).

- **Results**

  The results tab shows the results of endpoints that have been scanned, including both positive (true, a certificate was found) or negative (false, a certificate was not found) results. If a response was received from an endpoint during a scan, it is included in the results; negative results are hidden by default. The *Monitor Status* (True/False) and *Reviewed Status* (True/False) of an endpoint are included in the results tab.

The SSL network discovery and monitoring features can only be used if at least one compatible (see Compatibility Matrix) instance of the Keyfactor Universal Orchestrator is running in the environment and the orchestrator has been approved in the Management Portal. Keyfactor recommends that the orchestrator(s) used for SSL network discovery and monitoring be installed on a server other than the primary Keyfactor Command server(s) due to the resource requirements of the scanning process when scanning large network segments.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Network Definitions

On the Network Definitions tab you can create, edit and delete networks, and run and view scans. This section is also used to view the results of the discovery and monitoring jobs by linking directly to the Results tab for a selected network.

Discovery jobs attempt to initiate TLS connections to specified IP addresses and ports or ranges of IP addresses and ports. If a TLS connection is successful, the certificates provided by the target server as part of the TLS handshake are downloaded for further inspection and importation into the Keyfactor Command database. Locations that provide any level of response during the connection attempt (don't time out) are shown in the results grid when the discovery scan finishes regardless of whether a certificate was successfully downloaded. If a TCP connection is established, but a TLS connection is not, an SSL connection will be attempted. Any certificate obtained via SSL connection will be imported into the Keyfactor Command database. If a TLS connection is successful, an SSL connection will not be attempted.

Monitoring jobs scan a chosen set of locations that have already been discovered by a discovery job scan. Like discovery jobs, monitoring jobs attempt to initiate TLS connections with the locations specified. In the case of monitoring jobs, however, a certificate is expected at the endpoint since endpoints are generally identified for monitoring if they have certificates that need monitoring. As a result, monitoring jobs report on timeouts as well as connection failures and successes.

The network definitions grid includes these fields:

Name

The name of the network.

Orchestrator Pool

The name of the orchestrator pool (see Orchestrator Pools Definition on page 470).

Discovery Status

The current status of the discovery job for the network, if configured. The possible statuses are:

- *Scheduled* indicates a job has been scheduled but has never run.
- *Last Scanned* indicates a job has run to completion. The date indicates when the job finished.

Monitor Status

The current status of the monitoring job for the network, if configured. The possible statuses are the same as those for discovery.

Description

The description entered in the network definition.

- *Running* indicates a job is currently in progress. The percentage complete shown will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment).
- *In Quiet Hours* indicates that a quiet hours time window for the job is currently in effect and no jobs can be run (see SSL Network Operations below).
- *Disabled* indicates that the *Scanning Enabled* box has been unchecked in the network definition (see SSL Network Operations below). No scanning jobs will be run when the network is in this state.

SSL Network Operations

SSL Network Operations include adding, editing and deleting SSL network definitions, initiating a manual scan and monitoring scheduled network scan jobs.

Tip: SSL scan jobs use priority rules to determine which job segments run first if there are multiple job segments to be run (large jobs are divided into multiple job segments—see Monitoring Network Scan Jobs with View Scan Details on page 465). Job segments are run with the following priority rules:

- Job segments for Scan Now jobs (see Initiating a Manual Scan on page 466) are run ahead of those for scheduled jobs.
- New job segments for in-progress jobs with multiple segments are prioritized based on job age—segments for jobs that have been running the longest move to the front of the line.
- New job segments for in progress jobs with multiple segments start ahead of job segments for jobs that have not yet started.

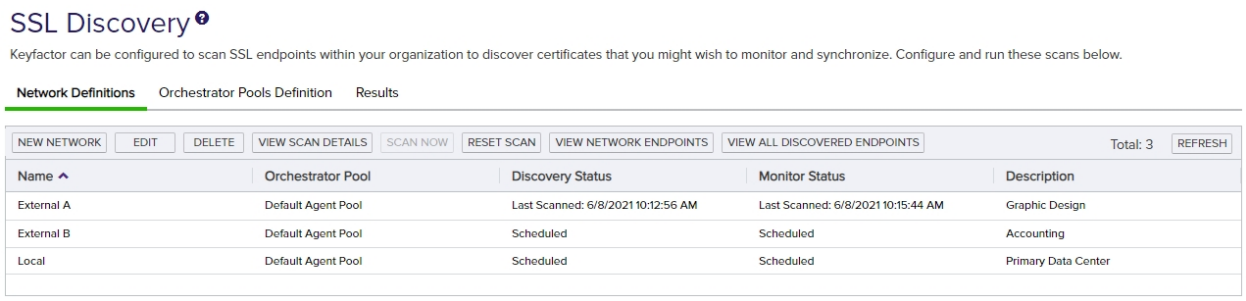


Figure 256: SSL Network Discovery

Adding or Modifying an SSL Network

To define a new network or edit an existing one:

1. In the Management Portal, browse to *Locations > SSL Discovery*.
2. On the SSL Network Discovery page, select the Network Definitions tab (the default when you first visit the page).
3. On the Network Definitions tab, click **New Network** to setup a network to scan, or select an existing network from the grid and click **Edit.**
4. The SSL Network Definition dialog is divided into four tabs: Basic, Advanced, Network Ranges, and Quiet Hours. Enter the network information for each tab, as required. Each tab is described in detail below.

Basic Tab

In the SSL Network Definition dialog on the Basic tab, enter the following information:

- **Name**: Enter a name for the network. The network name can be anything; however, it is recommended that the name reflect the subnet or location that you will be discovering with the network.

  Tip: The SSL network name is searchable with certificate search and also appears in the location details grid of the certificate details, if the certificate was found during an SSL scan.

- **Description**: Enter a description for the network.
- **Orchestrator Pool**: From the dropdown, select an orchestrator pool that contains orchestrators with SSL discovery and monitoring capabilities.

  Note: Keyfactor Command is installed with a Default Orchestrator Pool and orchestrators with SSL discovery and monitoring capabilities created in Keyfactor Command are automatically assigned to that pool.

- **Discovery/Monitoring Schedule**: Select the discovery and monitoring job frequency. Possible options are:
  - Off—No jobs will run.
  - Daily—Enter selected time.
  - Interval—Enter an interval from every 10 minutes to every 12 hours.
  - Weekly—Enter a selected day or days of the week at a selected time.
  - Monthly—Enter a selected day of the month (1st through 27th) at a selected time.

Note: The configured schedule determines when the scan is requested to start. The actual start of the scan is dependent on the orchestrator heartbeat Interval, which is defined by the *Heartbeat Interval (minutes)* application setting (see Application Settings on page 601). The default is 5 minutes.

- **Notification Recipients**: Enter one or more email address(es) of the recipients who should receive monitoring results (newline separated).

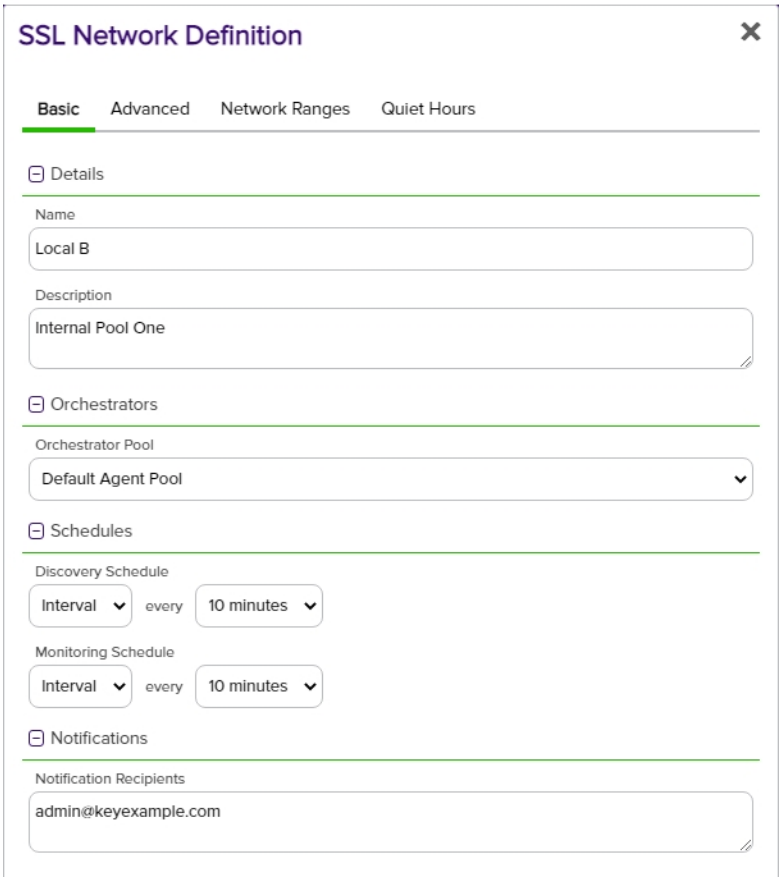


Figure 257: Define a New Network—Basic Tab

Advanced Tab

In the SSL Network Definition dialog on the Advanced tab, enter the following information:

- **Scanning Enabled** : Click to enable scanning for the network. If unchecked, no new network scans will be scheduled, but the current scan will finish, if this setting is changed during a scan also, the network will appear as *Disabled* on the SSL Network Discovery page.

- **Automatically monitor network endpoints during discovery**: Enable this option to instruct the orchestrator to tag endpoint certificates, found during discovery scanning, for monitoring. It is recommended to enable this option.
- **Request robots.txt**: Each network definition contains an option to do a GET on robots.txt on endpoints. Orchestrators perform a GET /robots.txt request to behave like a webcrawler and provide an explanation of network activity.
- **Discover Timeout (in ms)**: Enter the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however will also increase the chance of missing a certificate on a slow or congested network
- **Monitor Timeout (in ms)**: Enter the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
- **Expiration Alert (in days)**: Enter the number of days within which to begin warning regarding upcoming expiration in notification email messages.

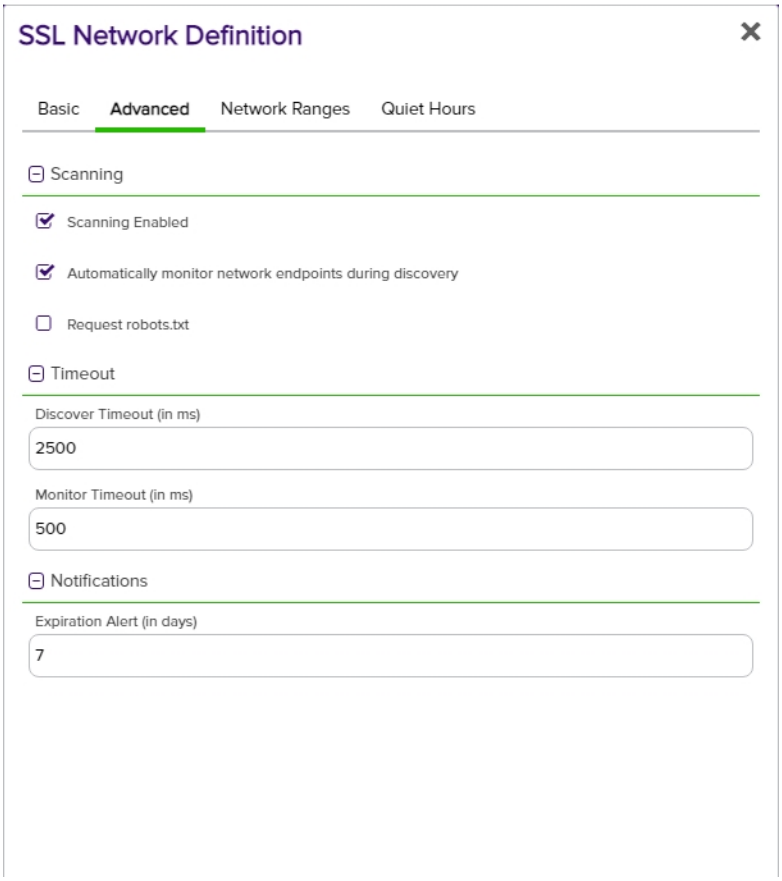


Figure 258: Define a New Network—Advanced Tab

Network Ranges Tab

There are two sections to the Network Ranges Tab: **Add Range** and **Ranges**. For each named network defined, multiple ranges are allowed. New networks can be added by using either the *add range tool*, or pasting the IP address, hostname, or network notation into the Network Ranges box.

The Add Range section

The **Add Range** section is for adding new networks via the *add range tool.* When you open the Network Ranges tab, the **Add Range** section shows default values of:

- **Type**: Network Notation
- **CIDR Block**: 0.0.0.0/24:443

Notice that the details grid reflects the default value and the default type—*network notation*. As you begin entry of a new network range of the type *network notation*, the details section

will reflect your entries as you type, allowing you to verify your entry. The details grid will not show if you chose another type of notation.

Define new network locations, using the add range tool, as follows:

a. In the **Add Range** section of the page, select your desired method for adding a location in the **Type** dropdown. The available options are:

   - **Network Notation**: Enter an IP address range using CIDR notation by populating the **CIDR Block** field and selecting the desired subnet in the dropdown. The default subnet is /24, which is one full octet of variability, or 254 locations.
   - **IP Address**: Enter a single IP address by populating the **IP Address** field and adding one port.
   - **Host Name**: Add a single location using a host machine name by filling in the **Host Name** field in the host name section and adding one port. During scans, host names are converted to IP addresses and scans are conducted via IP address. Keyfactor Command will do two scans against that address, one using the hostname as the SNI (server name indication) and one not using SNI. This is because different servers can be hosted on the same IP address but are accessed via different SNIs (or without one at all).

     Note: All methods support adding multiple ports, either comma separated (433,450), or as a range (433-450).

b. Enter the desired network notation, IP address, or host name, and click the **Add** action button.

c. Repeat this step for multiple IP addresses or host names. Each entry will be added as a newline in the Network Ranges box at the bottom of the dialog.

d. Click **Save**.

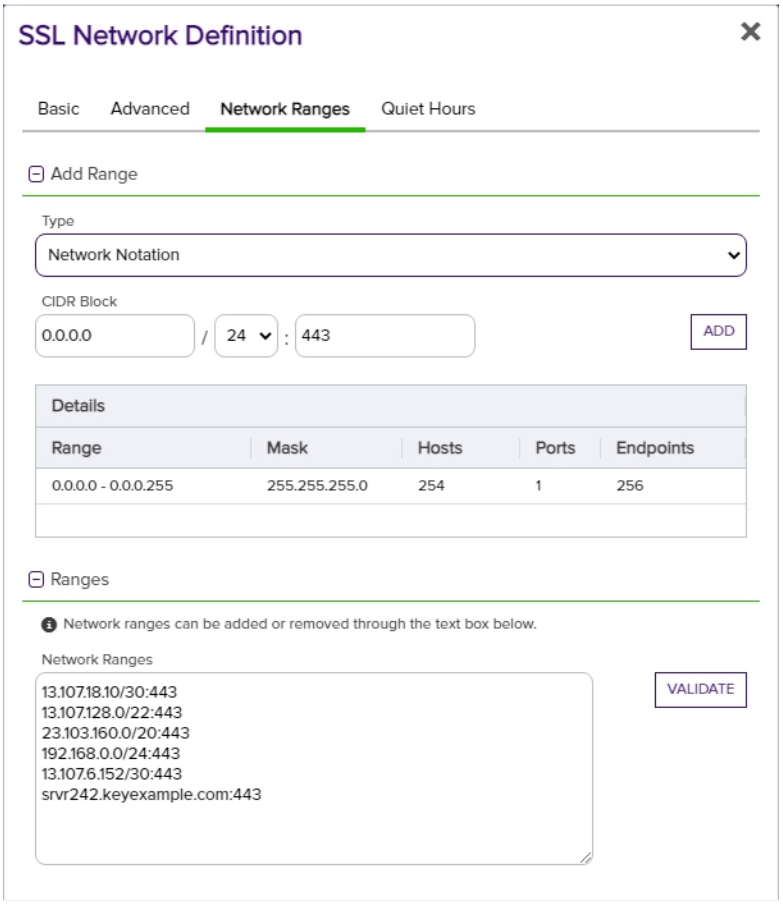


Figure 259: Define a New Network—Network Ranges Tab

The details grid displays only for the type *network notation* and will only display the value being typed in the CIDR block, or the last value entered. The fields in the details grid are defined as follows:

- **Range**: This is the range of addresses reflected by the CIDR notation entered.
- **Mask**: Defined by the bitmask (between 1 -30) applied to the address in the CIDR block to identify the IP addresses included. The bigger the mask the fewer IP addresses will fall under the defined range. For example, with a "/24", the first 3 sections of the IP address must match exactly, while the last section can be any value from 0 to 255.
- **Hosts**: This is the number of useable IP addresses in a given CIDR. (This is always two less than the number of endpoints. This is because the smallest address is reserved as

the address of the overall network the CIDR represents, while the largest is used as the broadcast address).

- **Ports**: This is the number of ports the given CIDR will have.
- **Endpoints**: The endpoints number reflects the number of endpoints based on the network size (/24, /25, etc) times the number of ports defined. Each time you go up in network size the network number will double (”/24” has 256, ”/23” has 512, ”/22” has 1,024 etc). So if you have just one port defined, the number of endpoints will be 256 for a ”/24” network, but if you had 3 ports (like say 443-445) that number would jump to 768. The same scenario for a ”/23” network would be 512 for one port and 1,536 for three ports.

The Ranges section

You can see any existing network definitions in the Network Ranges box in the **Ranges** section of the dialog. The **Ranges** section:

- Displays existing defined network ranges.
- Allows you to edit or delete existing network ranges. To delete a network range, highlight the selected range and click **Delete** on your keyboard. To edit a network range, highlight the selection to change and type over with the desired value(s).
- Accepts typed or pasted ranges, bypassing the *add range tool*. To add a network range, click inside the network ranges text box and type the desired value(s) or paste from your local clipboard. Ranges added this way must also contain the ports notation (e.g, :443).
- Validates network ranges as defined. To validate the list of ranges defined for the network, click the **Validate** action button. Based on the result, either a green *Network ranges are valid* message will display, or an alert will pop up with the list of invalid ranges.

Quiet Hours Tab

Quiet hours are ranges of hours or days during which scanning will not take place. Any scans in progress when the quiet hour window is reached will pause for the duration of the window and resume when the window is complete. SSL scans will show a status of **In Quiet Hours** if scanning is currently in that status.

In the SSL Network Definition dialog on the Quiet Hours tab, define quiet hour periods as follows:

a. In the Add Quiet Hours section of the page, select a day and time to begin a quiet hour period in the *Start* section.

b. Select a day of the week and time to end the quiet hour period in the *End* section.

c. Click **Add** to add the quiet hour period to the Quiet Hours section of the page.

d. Repeat the above steps for any additional quiet hour periods.

Note: Quiet hours replace and expand upon the blackout period option that existed in previous versions of Keyfactor Command.

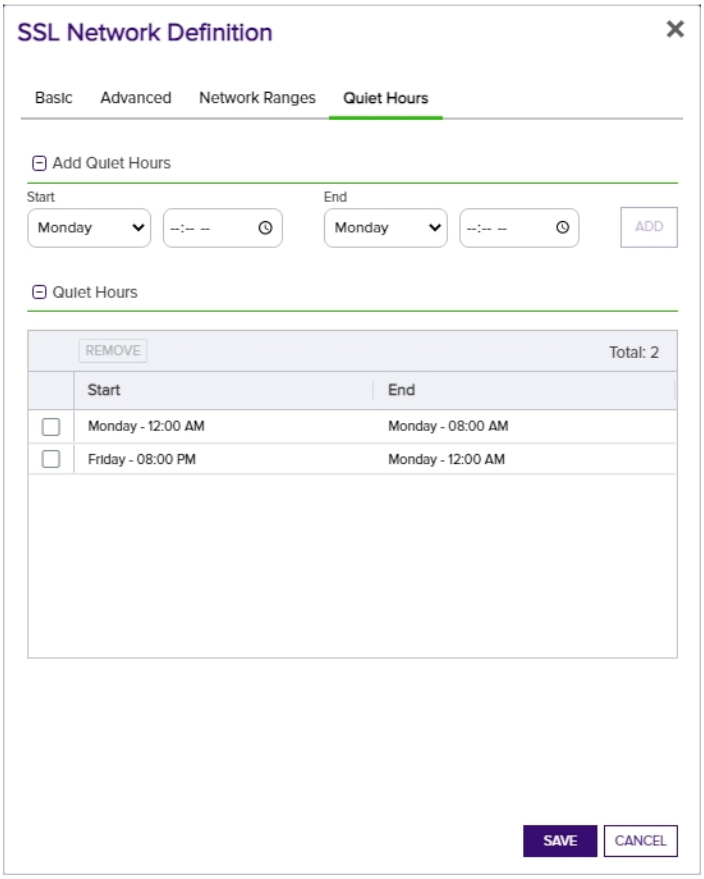


Figure 260: Define a New Network—Quiet Hours Tab

5. Click Save to save the new network definition or changes.

Deleting an SSL Network

1. In the Management Portal, browse to *Locations > SSL Discovery*.
2. On the SSL Network Discovery page, select the Network Definitions tab (the default when you first visit the page).

3. On the Network Definitions tab, highlight the row in the SSL network grid of the network to delete and click **Delete** at the top of the grid or right-click the network in the grid and choose **Delete** from the right-click menu.

4. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Monitoring Network Scan Jobs with View Scan Details

At any time, you can view the status of the latest scan jobs by viewing scan details from the *SSL Discovery* page. Right-click the network location in the grid and choose **View Scan Details** from the right-click menu or highlight the row in the network grid and click **View Scan Details** at the top of the grid.

This takes you to a separate page with separate tabs for Discovery and Monitoring jobs (see Figure 261: SSL Network Scan Details Page). Details for the last scanned job display above the grid in each tab and the scanned segments for the latest scan populate the grid. You will only see more than one row in the grid if the SSL management job was broken into segments due to having a large number of endpoints. The number of endpoints per segment is configurable (see the *SSL Maximum Scan Job Size* setting in Application Settings: Agents Tab on page 614). The grid will display the latest completed job and will be refreshed with new scan details when the next scan begins.

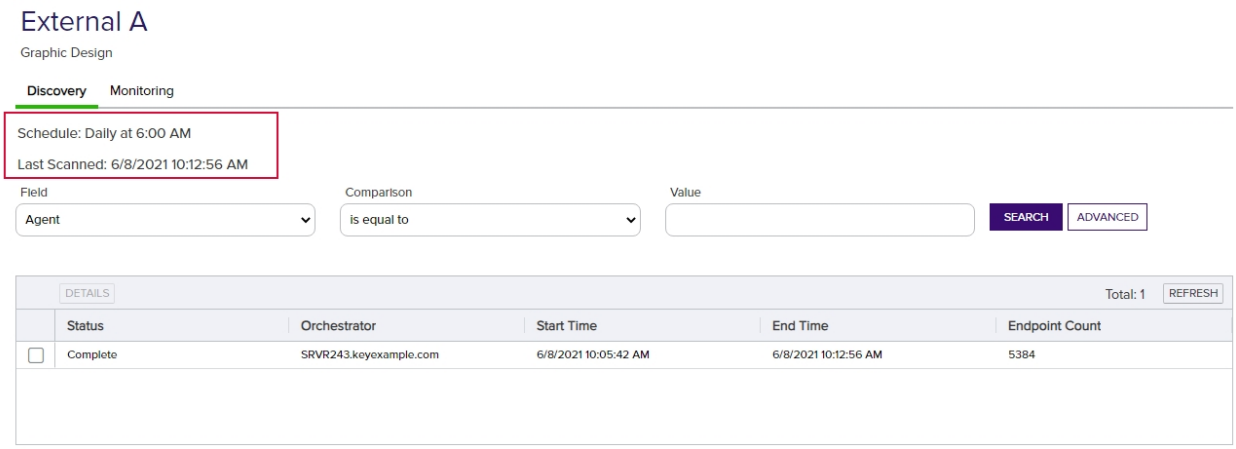


Figure 261: SSL Network Scan Details Page

To view details for a segment, double-click the segment, right-click the segment and choose **Details** from the right-click menu, or highlight the row in the scan details grid and click **Details** at the top of the grid (see SSL Network Scan Detail Segment Details on the next page).

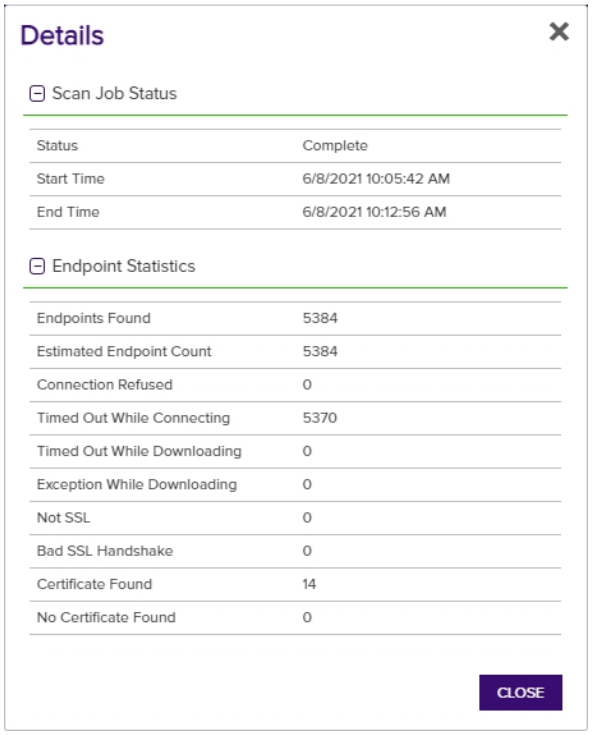


Figure 262: SSL Network Scan Detail Segment Details

Tip: If jobs are taking longer to complete than expected, see Slow SSL Jobs on page 829.

Initiating a Manual Scan

In addition to SSL scanning jobs that can be run as scheduled, the Network Definitions tab includes a feature that allows you to manually initiate a scan for a configured network at any time that a scan is not already running for the network or the network is not in quiet hours. When you initiate a scan using the scan now feature, you can choose whether to run a discovery scan, a monitoring scan, or both.

To initiate a manual scan for a network:

1. In the Management Portal, browse to *Locations > SSL Discovery*.
2. On the SSL Network Discovery page, select the Network Definitions tab (the default when you first visit the page).
3. On the Network Definitions tab, highlight the row in the SSL network grid of the network to scan and click **Scan Now** at the top of the grid or right-click the network in the grid and choose **Scan Now** from the right-click menu. The scan will begin immediately.

Tip: If a scan is already in progress for the network, the option to start a scan of that type will be grayed out and cannot be selected.

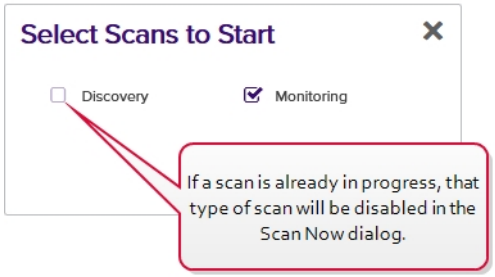


Figure 263: SSL Network ScanNow

Reset Scan

Resetting an SSL scan deletes all scan jobs, scan job parts, logical scan jobs, and current schedules associated with the selected network. The agent job status relating to the SSL scans is set to failed and completed, and the agent is forced to register for a new session. Afterward, *Scan Now* is enabled to allow you to initiate a manual scan. When you select *Reset Scan*, you will receive a **Confirm Operation** message. Click **OK** to proceed or **Cancel** to quit.

Tip: If you have an SSL scan job that appears stuck or crashed without a failure result, you can use the reset scan option to cancel the dysfunctional scan job.

View Network Endpoints and View Discovered Endpoints

See the Results on page 472 documentation for more information on these action buttons.

Using the Network Scan Details Search

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

Agent

Complete or partial matches with the orchestrator name as listed in the Orchestrator field.

Status

Status matches or doesn't match the selected category—Not Started, In Progress, Complete

The SSL scan will show a status of *In Quiet Hours* if scanning is currently in that status. See SSL Network Operations on page 456.

Start Time

The time at which scanning of the segment began. Supports the %TODAY% token (see Advanced Searches on the next page).

EndTime

The time at which scanning of the segment began. Supports the %TODAY% token (see Advanced Searches on the next page).

EndpointCount

The number of endpoints scanned in the segment. The maximum number of endpoints per segment is configurable (see the SSL Maximum Scan Job Size setting in Application Settings: Agents Tab on page 614).

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

In addition to the options available in the query builder, three special values can be used in selected searches by typing them in directly:

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a

domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).

- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

Important: The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

Orchestrator Pools Definition

SSL network discovery and monitoring scanning is performed by assigning an orchestrator pool, containing orchestrators with discovery and monitoring capabilities, to a network. An orchestrator pool contains one to many orchestrators that support the SSL discovery and monitoring capabilities. Network scanning using orchestrator pools allows the work to be dispersed among the orchestrators in the pool.

Out of the box, all approved Windows orchestrators and Keyfactor Universal Orchestrators with the SSL capability are assigned to a default orchestrator pool. For scanning of larger and more complicated networks, orchestrator pools can be configured with multiple orchestrators running concurrently to perform the scanning operation.

Note: Approved orchestrators assigned to a custom pool will be removed from the default orchestrator pool. If a custom pool is removed, the orchestrator will be re-assigned to the default orchestrator pool.

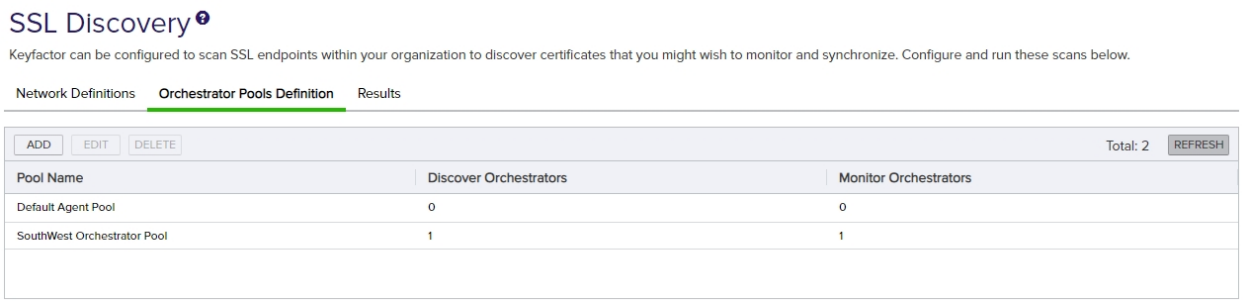


Figure 264: SSL Orchestrator Pools

Orchestrator Pool Operations

Orchestrator pool operations include: creating, editing or deleting pools.

Adding or Modifying an Orchestrator Pool

1. In the Management Portal, browse to *Locations > SSL Discovery*.
2. On the SSL Network Discovery page, select the **Orchestrator Pools Definition** tab.
3. On the Orchestrator Pools Definition tab, click **Add** from the top menu to create a new pool, or **Edit** from either the top or right click menu, to modify an existing one.

   Note: The available edit options include edit the name of the pool or select/de-select the discover/monitor options.

4. In the SSL Orchestrator Pool Definition dialog, enter a unique Orchestrator Pool name in the **Name** field.

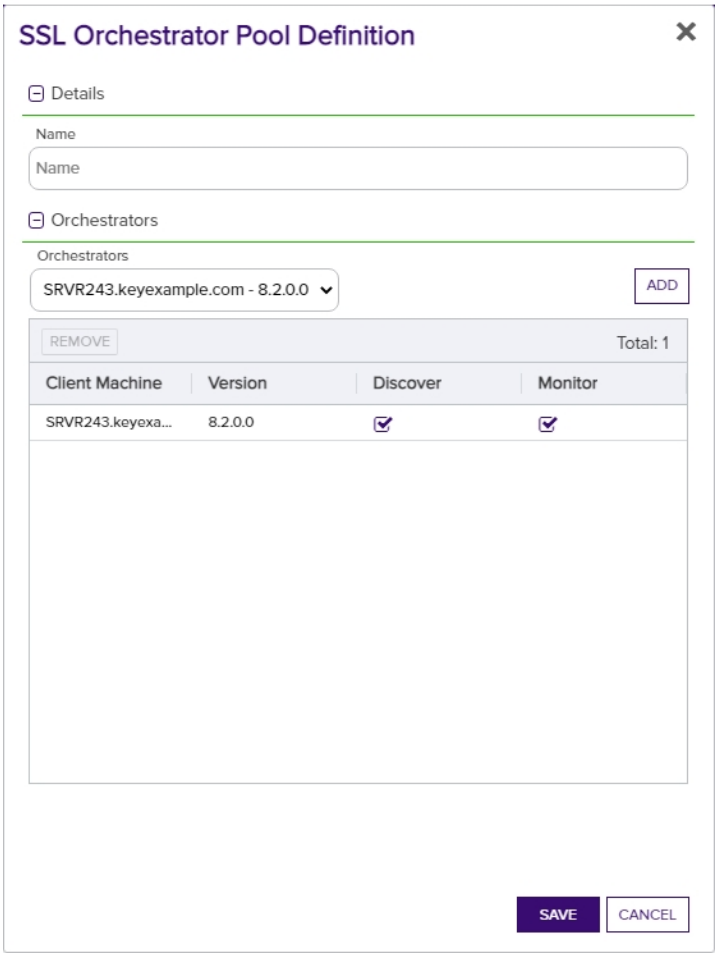


Figure 265: Add an Orchestrator Pool

5. From the **Orchestrators** dropdown, select eligible orchestrators—those orchestrators that support monitor and discovery capabilities—to add to the orchestrator pool and click **Add**.

   Tip: Orchestrators are added with discover and monitor responsibilities. You can de-select one of these options, if needed.

6. Highlight a row and click **Remove** to remove the orchestrator from the orchestrator pool. The orchestrator will be returned to the default orchestrator pool.

   Note: You are not able to remove orchestrators from the default orchestrator pool; they are automatically removed if assigned to a custom orchestrator pool.

7. Click **Save** to save the orchestrator pool.

Deleting an Orchestrator Pool

You may delete one expiration record at a time.

1. In the Management Portal, browse to *Locations > SSL Discovery*.
2. On the SSL Network Discovery page, select the **Orchestrator Pools Definition** tab and select the row you wish to delete.
3. Click **Delete** at the top of the grid, or from the right click menu.
4. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

   Note: You are not able to remove the default orchestrator pool.

Results

The SSL network discovery and monitoring results include endpoints that returned certificates as well as endpoints that resulted in some level of response (did not time out) but did not return certificates.

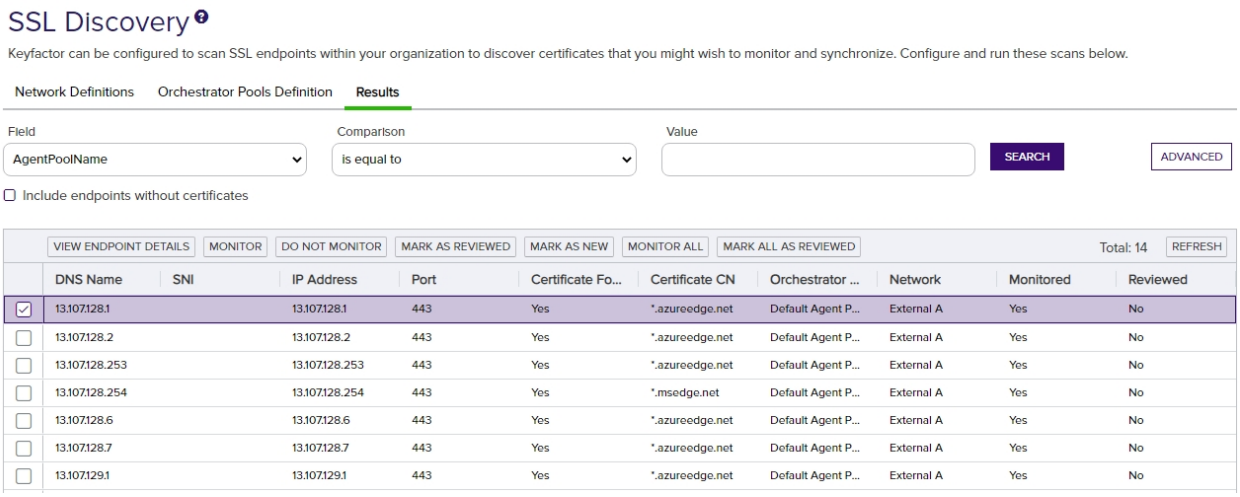


Figure 266: SSL Discovery Results

For each endpoint discovered during the scan, the results grid includes the following:

DNS Name

The host name converted to an IP address, or the IP address scanned. The DNS name is resolved by the orchestrator performing the scan, based on the DNS settings of the server running the orchestrator.

SNI

The server name indication (SNI), if one is found.

IP Address

The IP address scanned.

Port

The port scanned.

Certificate Found

Whether a certificate was found at the endpoint on the most recent scan (true/false).

Certificate CN

Common name discovered on the certificate.

Orchestrator Pool

The orchestrator pool name that contains the orchestrator that discovered and/or monitored the endpoint.

Network

The name of the network.

Monitored

Whether the discovered endpoint is configured for monitoring (true/false). If the *Automatically monitor endpoints found during discovery* option is enabled in the network definition, the orchestrator will, upon initial discovery, monitor the discovered certificate. You can change the monitoring status of a discovered endpoint in the results grid.

Reviewed

The discovered endpoint has been reviewed (true/false). To denote an endpoint as reviewed, highlight the row in the results grid and click **Mark as Reviewed** at the top of the grid or right-click the endpoint and choose **Mark as Reviewed**.

Using the Discovery Results Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

AgentPoolName

Complete or partial matches with the orchestrator pool that contains orchestrators used to discover and monitor the results.

CertificateCN

Complete or partial matches with the certificate common name.

NetworkName

Complete or partial matches with the network name.

Port

Numeric matches with the port number for the discovered endpoint.

CertificateFound

Certificate was found at the endpoint on the most recent scan (true/false).

ReverseDNS

Complete or partial matches with the DNS name resolved based on the discovered IP address. If a host name could not be resolved, this will be the IP address.

IPAddress

Complete or partial matches with the IP address.

IsMonitored

Endpoint has been marked as monitored (true/-false). By default, only endpoints that are marked as monitored equals true are displayed.

Issuer DN

Complete or partial matches with the issuer distinguished name.

Reviewed

Whether it is true or false that the scan has been reviewed.

SelfSigned

Certificate is self-signed (true/false).

SNIName

The server name indication (SNI) of the endpoint.

Status

The status of the scan. Options include:

- Certificate Found
- Timed Out Connecting
- Exception Connecting
- Timed Out Downloading
- Exception Downloading
- Not SSL
- Exception in Sql
- Invalid or Unreachable Host
- Connection Refused
- Bad SSL Handshake
- Client Authentication Failed
- No Certificate
- SSL Refused
- Not Probed
- Unknown

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template

referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

SSL Results Action Options

The following options are available on the results tab:

- To view the details for a discovered endpoint, double-click the result, right-click the result and choose **View Endpoint Details** from the right-click menu, or highlight the row in the results grid and click **View Endpoint Details** at the top of the grid (see Viewing Endpoint Details below).
- To add a discovered endpoint to a monitoring job, right-click the result and choose **Monitor** from the right-click menu, or highlight the row in the results grid and click **Monitor** at the top of the grid.
- To remove an endpoint from a monitoring job, right-click a result that has a Monitor Status of *true* and choose **Do Not Monitor** from the right-click menu, or highlight the row in the results grid and click **Do Not Monitor** at the top of the grid.
- To change endpoints to reviewed, right-click the result and choose **Mark as Reviewed** from the right-click menu, or highlight the row in the results grid and click **Mark as Reviewed** at the top of the grid. Newly found endpoints default to a reviewed state of *false*.
- To change reviewed endpoints to not reviewed, right-click the result and choose **Mark as New** from the right-click menu, or highlight the row in the results grid and click **Mark as New** at the top of the grid.
- To add all discovered endpoints to a monitoring job, click **Monitor All** at the top of the grid.
- To change all endpoints to reviewed, click **Mark All as Reviewed** at the top of the grid.
- You can click the **Include Endpoints without Certificates** button at the top of the results grid to toggle inclusion of endpoints without certificates in the results. By default they are excluded.

To select a single row in the grid, click to highlight it and then select an operation from either the top of the grid or the right-click menu. Some of the operations support action on multiple results at once. To select multiple rows, hold down the CTRL key and click each row on which you would like to perform an operation. Then select an operation from the top of the grid. The right-click menu supports operations on only one certificate at a time.

Viewing Endpoint Details

To view details of the scan history and certificates found for an SSL job, in the SSL discovery results grid, double-click the result, right-click the result and choose **View Endpoint Details** from the right-click menu, or highlight the row in the results grid and click **View Endpoint Details** at the top of the grid. The endpoint history dialog includes this information:

- The **SSL/TLS Endpoint Details** section of the dialog includes details of the selected certificate including the IP address, port, DNS name, SNI (if available), endpoint network name, orchestrator pool name that contained the orchestrator which performed the scan, and monitoring status of the certificate (true/false).

- The **Chain Level** dropdown allows you to view details of certificates chained to the selected certificate. The default is the end entity certificate.
- The **Certificate Details** section provides details of the certificate selected in the chain level dropdown.
- The **Endpoint History** section on the right side of the endpoint history dialog details each individual scan including the date of the scan, source (monitoring or discovery), the IP address and the certificate status.

The details menu can also provide information on why a certificate was not found if one was expected.

Endpoint history records on the endpoint details page older than 30 days, by default, are automatically purged daily. You can change the length of time for which records are retained by updating the *Retain SSL Endpoint History (days)* in the application settings.

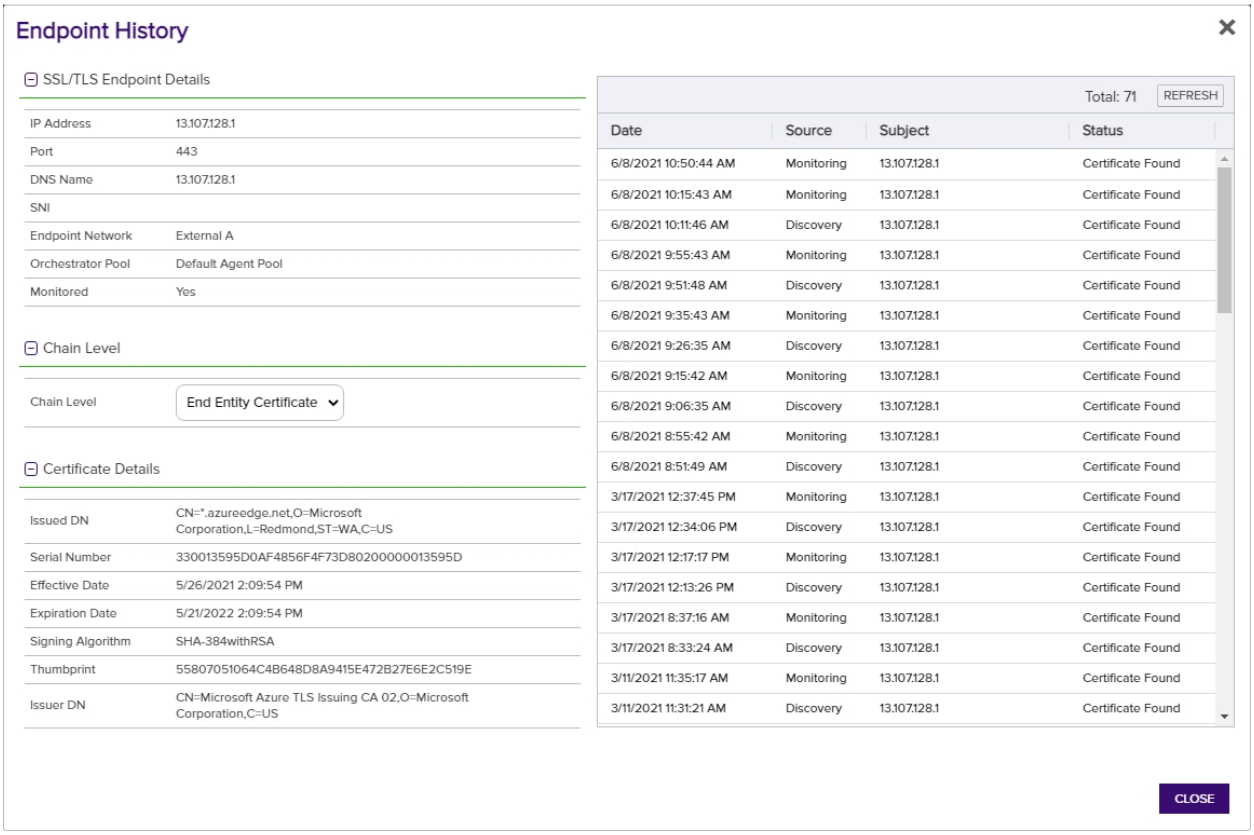


Figure 267: SSL Discovery and Monitoring Result Details

Tracking the Expiration on SSL Certificates

At the conclusion of a of a network monitoring scan, an email is sent to the configured recipients indicating which, if any, certificates associated with that network are nearing expiration. *Near* expir-

ation is determined based on the *Expiration Alert (in days)* setting in the network definition, which defaults to 7 days (see SSL Network Operations on page 456).

This is one method of tracking expiration on SSL certificates, but since the certificates are synchronized to the Keyfactor Command database, you can also use regular expiration alerts (see Expiration Alerts on page 167) and reports (see Reports on page 91) to track expiration for these certificates as you would for certificates issued from internal CAs.

Understanding Notification Emails

The discovery and monitoring notification emails that are delivered at the conclusion of discovery and monitoring scans both include information about the status of the endpoints scanned, but they present this information slightly differently. The discovery email breaks down what happened when the job attempted to find a certificate at each of the endpoints it attempted to communicate with. The monitoring email, on the other hand, focuses on monitoring the status of the certificate that is expected to be at the endpoint. Although the monitoring email can be used for identifying certificates that are coming up for expiration, other solutions, such as expiration alerts (see Expiration Alerts on page 167), may be more useful for this. What the expiration alerts can't do for you, however, and the monitoring email can, is identify servers that may have gone offline or whose certificate may have disappeared. In other words, expiration alerts monitor certificate status and monitoring alerts monitor endpoint status. See the example in Figure 269: SSL Monitoring Email. This shows three servers that previously had been discovered to have a certificate now being unresponsive. In some cases, the servers or certificates may still be there and the requests for them have just timed out due to slow network connections or other issues, but this provides you with an opportunity to investigate these servers to determine what the problem might be.

The various numbers that are reported in the Discovery and Monitoring emails are:

- **The number in the subject:** The total number of endpoints that have expired/expiring certificates plus the total number of endpoints that did not return a certificate.
- **Expired/Expiring certificates number:** The total number of certificates that are expired or will expire within the next X number of days. The value of X is a configurable setting in Keyfactor Command and is set in the network definition for each network (see the *Expiration Alert* setting in SSL Network Operations on page 456).
- **Number of endpoints that did not return a certificate**: The total number of endpoints that did not return a certificate.
- **Number of rows in each grid:** A configurable setting in Keyfactor Command (see the *SSL Maximum Email Results* application setting in Application Settings: Agents Tab on page 614). The number of rows in the grids is not reflected in the total counts.

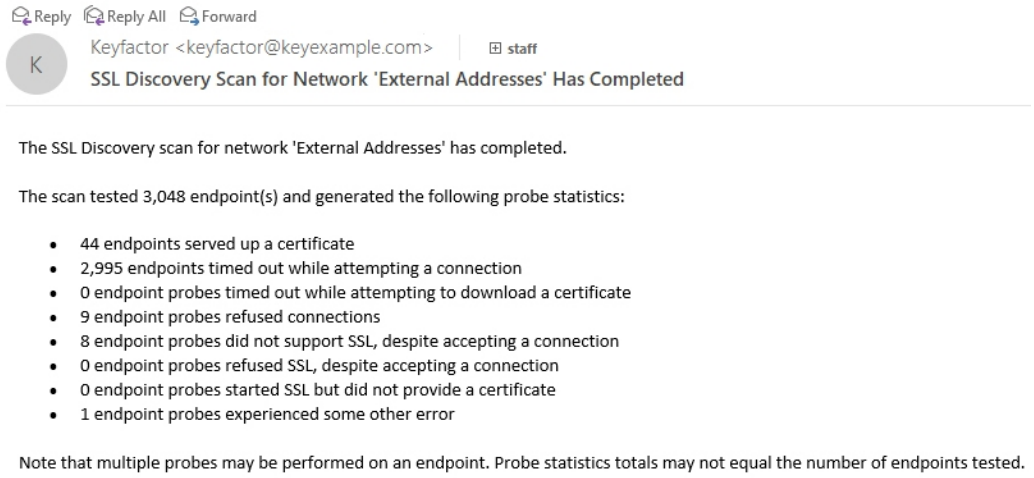


Figure 268: SSL Discovery Email

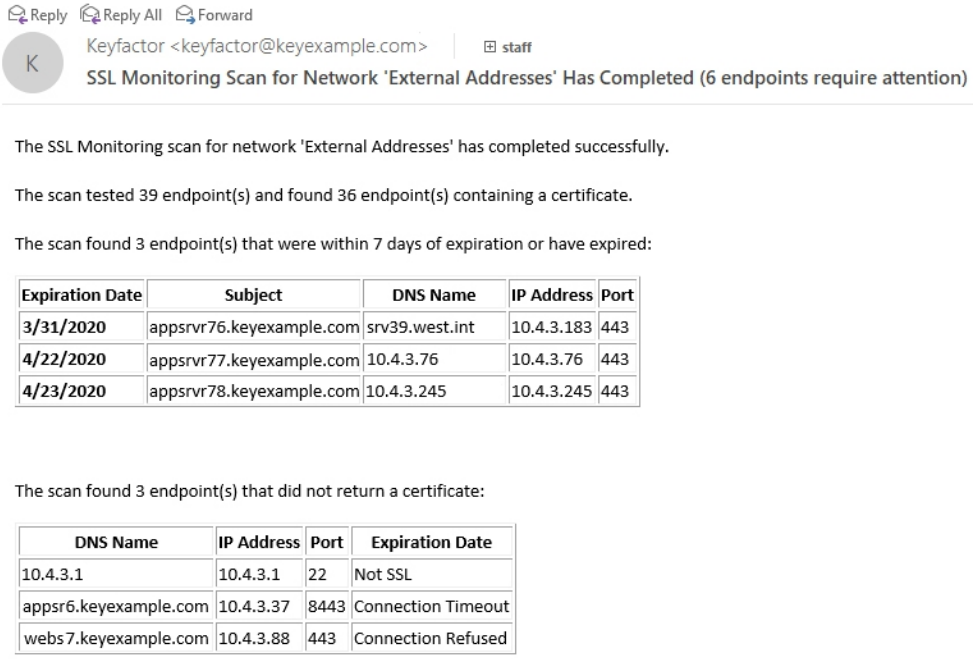


Figure 269: SSL Monitoring Email

Table 15: SSL Email Notification Values Defined

Value	Meaning
Timed out while	A timeout occurred when attempting to establish a TCP connection. The timeout

Value	Meaning
connecting	interval is defined on the Advanced tab of the SSL network definition page, see SSL Network Operations on page 456. The shorter the timeout, the faster the scan goes, but the higher chance that if there is actually something listening at the port, a connection won't be established causing a timeout. If the orchestrator is overloaded (too many parallel tasks), it can add to the time needed to make a connection and increase the chance of a timeout. Network transit time affects timeouts as does the load and speed of the target system in the ability to establish a TCP handshake.
Timed out while downloading	A TCP connection was made and a TLS connection was started, but it took too long to actually receive the certificate. This is a rare condition. This is a parameter that is locally configurable on the orchestrator and defaults to 15 seconds. This value is displayed in the debug trace.
Connection refused	The target IP and Port are listening, but the TCP connection was actively refused.
Not SSL	A TCP connection was established, but when the first packet of the TLS handshake was sent, it did not get a TLS response, implying that some protocol other than TLS is listening on the target.
Bad SSL handshake	A TCP connection was established and a proper response to the first TLS packet was returned, but something failed in the rest of the TLS handshake. Several of the internal reasons for why a TLS handshake may have failed have been combined along with other counters in the email response.
Certificate found	A TCP connection and a TLS handshake were completed and the TLS handshake returned a certificate (all within the connection and download timeout periods)

2.1.9 Orchestrators

Keyfactor Command uses orchestrators (a.k.a. agents) to manage a wide variety of certificate store types. As of this writing, Keyfactor offers these orchestrators:

Keyfactor Universal Orchestrator

This orchestrator runs on Windows servers or Linux servers and is used to run jobs at the request of the Keyfactor Command server. Jobs primarily perform certificate management tasks, but other types of operations are also supported. Jobs are provided to the orchestrator as extensions; both built-in and custom extensions are supported. The orchestrator includes built-in extensions to run SSL discovery and management tasks, manage synchronization of certificate authorities in remote forests, and retrieve the orchestrator logs for analysis with the Keyfactor API.

Keyfactor Java Agent

This orchestrator runs on Windows or Linux servers and is used to perform discovery of Java keystores and PEM certificate stores, to inventory discovered stores, and to push certificates out to stores as needed.

Important: The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

https://github.com/Keyfactor/remote-file-orchestrator

For more information, see Installing Custom-Built Extensions on page 2940.

Keyfactor Mac Auto-Enrollment Agent

This orchestrator runs on Apple Macintosh computers and allows users to auto-enroll for certificates.

Keyfactor Android Agent

This orchestrator runs on Android OS Devices and is used to manage PEM and Java keystores. The orchestrator is distributed as part of the Keyfactor Integration SDK (software development kit). Contact Keyfactor for more information.

Keyfactor Native Agent

This orchestrator is a reference implementation intended for customers wanting to include Keyfactor Command certificate store management functionality in embedded or other platforms. The orchestrator is distributed as part of the Keyfactor Integration SDK (software development kit). Contact Keyfactor for more information.

Keyfactor AnyAgent

The Keyfactor AnyAgent runs on Windows or Linux servers and is used to allow management of certificates regardless of source or location by allowing customers to implement custom agent functionality. Custom store types and/or job capabilities, on which agents operate, are created by adding commands and leveraging extendable code to communicate through an API with Keyfactor Command. Because of the custom nature of the functionality of the AnyAgent, it is not included in the table below, as it could be designed to do one or more of the capacities below, or additional capacities not included below. Contact Keyfactor for more information.

Keyfactor Bash Orchestrator

This orchestrator runs on Linux servers and is used to perform discovery of SSH keys, generation of SSH keys, and management of SSH keys and Linux logons.

Table 16: Orchestrator Capabilities

	Universal	Java	Android	Native	Mac	Bash
Amazon Web Services Add/Remove	✓[1]					
Amazon Web Services Inventory	✓[1]					
Certificate Auto-enrollment					✓	
Certificate Reenrollment		✓	✓	✓		
Certificate Renewal	✓	✓	✓	✓		
F5 (Web Server, SSL Profiles, CA Bundles) Add/Remove	✓[1]					
F5 (Web Server & SSL Profiles, CA Bundles) Inventory	✓[1]					
F5 (SSL Profiles & CA Bundles) Discovery	✓[1]					
File Transfer Protocol Add/Remove	✓					
File Transfer Protocol Inventory	✓					
IIS (Personal, Revoked, Trusted) Add/Remove	✓[1]					
IIS (Personal,	✓[1]					

[1]Support for this functionality on the Keyfactor Universal Orchestrator requires the addition of a custom extension. For more information, see Installing Custom-Built Extensions on page 2940.

	Universal	Java	Android	Native	Mac	Bash
Revoked, Trusted) Inventory						
Java Keystore Add/Remove	✓[1]	✓	✓			
Java Keystore Create	✓[1]	✓	✓			
Java Keystore Discovery	✓[1]	✓				
Java Keystore Inventory	✓[1]	✓	✓			
Linux Logon Management						✓
Log Fetching	✓			✓		
NetScaler Add/Remove	✓[1]					
NetScaler Inventory	✓[1]					
PEM Add/Remove	✓[1]	✓	✓	✓		
PEM Discovery	✓[1]	✓				
PEM Inventory	✓[1]	✓	✓	✓		
Remote CA & Template Synchronization	✓					
SSL Discovery & Monitoring	✓					
SSH Key Discovery						✓
SSH Key Generation						✓

	Universal	Java	Android	Native	Mac	Bash
SSH Key Management						✔

The options available in the Orchestrator Management section of the Management Portal are:

Auto-Registration

Configure Keyfactor Command to allow orchestrators to auto-register.

Management

View and configure orchestrators.

Jobs

View active orchestrator jobs and review job errors.

Blueprints

Snapshot the certificate stores and scheduled jobs of one machine and apply them to multiple other similar machines.

Mac Auto-Enrollment

Configure settings for Mac auto-enrollment.

2.1.9.1 Orchestrator Auto-Registration

Orchestrator auto-registration allows you to automatically approve or deny new orchestrators without administrator input, if desired. This is useful in environments hosting a large number of orchestrators. On the Orchestrator Auto-Registration Settings page you define the conditions under which an orchestrator (e.g. Keyfactor Universal Orchestrator or Keyfactor Java Agent) can automatically be approved using the built-in auto-registration system. This is one of two ways that Keyfactor Command supports orchestrator auto-registration. Keyfactor Command also offers an enhanced orchestrator auto-registration system that allows the construction of custom orchestrator auto-approval handler modules. Any custom auto-registration handlers are processed first before the built-in auto-registration system runs. For more information about custom auto-registration handlers, see Custom Auto-Registration Handlers on page 495.

The configurable settings for the built-in auto-registration system are:

- Auto-Register

  Should orchestrators be allowed to auto register? If the *Auto-Register* box is checked but the *Validate Users* setting is not checked, any orchestrator that appears in your environment will automatically be approved regardless of origin.

- Validate Users

  Do the user accounts under which the orchestrators are running need to be a member of a specific group in order to auto-register (aka validation)?

  - User Groups

    If the user accounts must be a member of a group to auto-register (*Validate Users* is checked), which group or groups is that (or which user account if all orchestrators will be registering as the same user)? If the *Auto-Register* setting and the *Validate Users* settings are both enabled, then this field will be considered. If *Validate Users* is not checked, this setting will not be displayed.

The default auto-registration settings are to allow no orchestrators to auto-register.

Important: Orchestrator auto-registration in Keyfactor Command is only supported when using Active Directory as an identity provider (see Selecting an Identity Provider for Keyfactor Command on page 2704). If you need auto-registration with an identity provider other than Active Directory, see Custom Auto-Registration Handlers on page 495.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Orchestrator Auto-Registration Settings

Important: Orchestrator auto-registration in Keyfactor Command is only supported when using Active Directory as an identity provider (see Selecting an Identity Provider for Keyfactor Command on page 2704). If you need auto-registration with an identity provider other than Active Directory, see Custom Auto-Registration Handlers on page 495.

The Orchestrator Auto-Registration Settings grid shows the current auto-registration settings for any defined certificate store job types, including those for Keyfactor custom-built extensions. The list you will see in your implementation will vary and may include:

Amazon Web Services Inventory

Auto-register the Keyfactor Windows Orchestrator to allow it to synchronize certificates from AWS locations.

This certificate store type has been deprecated and is no longer in use.

Amazon Web Services Management

Auto-register the Keyfactor Windows Orchestrator to allow it to manage certificates on and deliver certificates to AWS locations.

This certificate store type has been deprecated and is no longer in use.

CitrixAdc Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to synchronize certificates from Citrix NetScaler devices.

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

CitrixAdc Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage certificates on and deliver certificates to Citrix NetScaler devices.

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

F5 CA Bundles REST Discovery

Auto-register the Keyfactor Universal Orchestrator to allow it to run discovery tasks to locate CA bundles on the F5 device(s).

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

F5 CA Bundles REST Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to synchronize CA bundles from the F5 device(s).

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

F5 CA Bundles REST Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage CA bundles on and deliver certificates to CA bundles on the F5 device(s).

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

F5 Certificate Store Inventory

Auto-register the Keyfactor Windows Orchestrator to allow it to synchronize certificates from the F5 device(s).

This certificate store type has been deprecated and is no longer in use.

F5 Certificate Store Management

Auto-register the Keyfactor Windows Orchestrator to allow it to manage certificates on and deliver certificates to the F5 device(s).

This certificate store type has been deprecated and is no longer in use.

F5 Keygen/re-enrollment

Setting reserved for future use.

This certificate store type has been deprecated and is no longer in use.

F5 SSL Profiles REST Discovery

Auto-register the Keyfactor Universal Orchestrator to allow it to run discovery tasks to locate SSL certificates on the F5 device(s).

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

F5 SSL Profiles REST Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to synchronize SSL certificates from the F5 device(s).

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

F5 SSL Profiles REST Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage SSL certificates on and deliver certificates to the F5 device(s).

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

F5 Web Server REST Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to synchronize device certificates from the F5 device(s).

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

F5 Web Server REST Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage device certificates on and deliver certificates to the F5 device(s).

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

File Transfer Protocol Inventory

Auto-register the Keyfactor Windows Orchestrator or Keyfactor Universal Orchestrator to allow it to synchronize certificates from PEM certificate stores on FTP capable devices.

This certificate store type has been deprecated and is no longer in use.

File Transfer Protocol Management

Auto-register the Keyfactor Windows Orchestrator or Keyfactor Universal Orchestrator to allow it to manage certificates on and deliver certificates to PEM certificate stores on FTP capable devices.

This certificate store type has been deprecated and is no longer in use.

IIS Bound Certificate Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to synchronize certificates from the machine certificate stores of Windows servers.

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

IIS Bound Certificate Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage certificates in and deliver certificates to the machine certificate stores of Windows servers and optionally bind the certificates to Internet Information Services (IIS) web sites.

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

IIS Bound Certificate Reenrollment

Auto-register the Keyfactor Universal Orchestrator to allow it to re-enroll for certificates in and deliver certificates to the machine certificate stores of Windows servers and optionally bind the certificates to Internet Information Services (IIS) web sites.

This type is user-defined for a Keyfactor custom-built extension publicly available on GitHub (see Installing Custom-Built Extensions on page 2940).

IIS Certificate Store Inventory

Auto-register the Keyfactor Windows Orchestrator or Keyfactor Universal Orchestrator to allow it to synchronize certificates from the machine certificate stores of Windows servers.

This certificate store type has been deprecated and is no longer in use.

IIS Certificate Store Management

Auto-register the Keyfactor Windows Orchestrator or Keyfactor Universal Orchestrator to allow it to manage certificates in and deliver certificates to the machine certificate stores of Windows servers and optionally bind the certificates to Internet Information Services (IIS) web sites.

This certificate store type has been deprecated and is no longer in use.

IIS Keygen/re-enrollment

This certificate store type has been deprecated and is no longer in use.

Java Keystore Discovery

Auto-register the Keyfactor Java Agent to allow it to run discovery tasks to locate Java keystores.

Java Keystore Inventory

Auto-register the Keyfactor Java Agent to allow it to inventory certificates in Java keystores.

Java Keystore Keygen/re-enrollment

This certificate store type has been deprecated and is no longer in use.

Java Keystore Management

Auto-register the Keyfactor Java Agent to allow it to manage (add/remove) certificates in Java keystores.

Mac Auto-Enrollment

Auto-register users on Apple Macintosh computers running the Keyfactor Mac Auto-Enrollment Agent for auto-enrollment for certificates.

NetScaler Certificate Store Inventory

Auto-register the Keyfactor Windows Orchestrator to allow it to synchronize certificates from NetScaler devices.

This certificate store type has been deprecated and is no longer in use.

NetScaler Certificate Store Management

Auto-register the Keyfactor Windows Orchestrator to allow it to manage certificates on and deliver certificates to NetScaler devices.

This certificate store type has been deprecated and is no longer in use.

NetScaler Keygen/re-enrollment

This certificate store type has been deprecated and is no longer in use.

Orchestrator Log Retrieval

Auto-register the Keyfactor Universal Orchestrator or Native Agent to allow it to perform the fetch logs function.

PEM Certificate Store Discovery

Auto-register the Keyfactor Java Agent to allow it to run discovery tasks to locate PEM certificate stores. Apache servers typically use PEM certificate stores.

PEM Certificate Store Inventory

Auto-register the Keyfactor Java Agent to allow it to inventory certificates in PEM certificate stores.

PEM Certificate Store Management

Auto-register the Keyfactor Java Agent to allow it to manage (add/remove) certificates in PEM certificate stores.

PEM Keygen/re-enrollment

This certificate store type has been deprecated and is no longer in use.

Remote Certificate Authority

Auto-register the Keyfactor Universal Orchestrator to allow it to synchronize certificates from the remote CA(s) to the Keyfactor Command database.

Remote File DER Discovery

Auto-register the Keyfactor Universal Orchestrator to allow it to run discovery tasks to locate DER certificate stores.

Remote File DER Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to inventory certificates in DER certificate stores.

Remote File DER Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage (add/remove) certificates in DER certificate stores.

Remote File JKS Discovery

Auto-register the Keyfactor Universal Orchestrator to allow it to run discovery tasks to locate JKS certificate stores.

Remote File JKS Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to inventory certificates in JKS certificate stores.

Remote File JKS Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage (add/remove) certificates in JKS certificate stores.

Remote File PEM Discovery

Auto-register the Keyfactor Universal Orchestrator to allow it to run discovery tasks to locate PEM certificate stores.

Remote File PEM Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to inventory certificates in PEM certificate stores.

Remote File PEM Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage (add/remove) certificates in PEM certificate stores.

Remote File PKCS#12 Discovery

Auto-register the Keyfactor Universal Orchestrator to allow it to run discovery tasks to locate PKCS#12 certificate stores.

Remote File PKCS#12 Inventory

Auto-register the Keyfactor Universal Orchestrator to allow it to inventory certificates in PKCS#12 certificate stores.

Remote File PKCS#12 Management

Auto-register the Keyfactor Universal Orchestrator to allow it to manage (add/remove) certificates in PKCS#12 certificate stores.

Remote Template Sync

Auto-register the Keyfactor Universal Orchestrator to allow it to synchronize templates from the remote CA(s) to the Keyfactor Command database.

Secure Shell Management

Auto-register the Keyfactor Bash Orchestrator to allow it to run SSH tasks.

SSL Endpoint Compliance

Auto-register the Keyfactor Universal Orchestrator to allow it to run SSL compliance tasks.

SSL Endpoint Discovery

Auto-register the Keyfactor Universal Orchestrator to allow it to run SSL discovery tasks.

SSL Endpoint Monitoring

Auto-register the Keyfactor Universal Orchestrator to allow it to run SSL monitoring tasks.

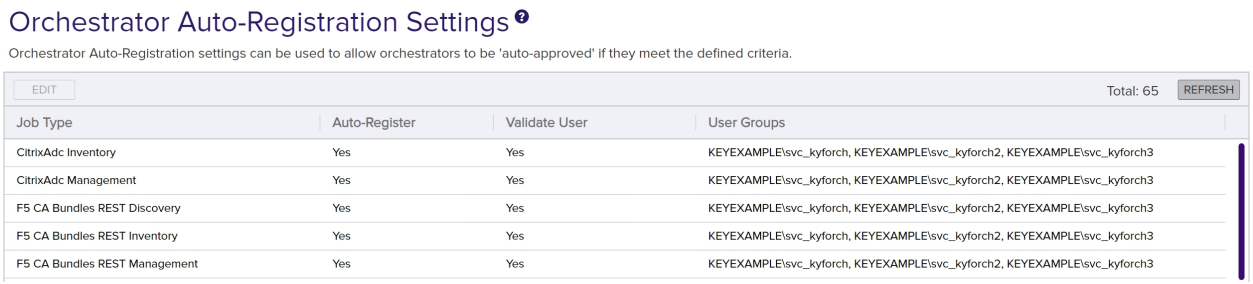


Figure 270: Orchestrator Auto-Registration Settings Page

Editing Orchestrator Auto-Registration Jobs

To edit one of the orchestrator job types:

1. In the Management Portal, browse to *Orchestrators > Auto-Registration*.
2. On the Orchestrator Auto-Registration Settings page, highlight the row in the grid of the job you want to edit and click **Edit** at the top of the grid or right-click the job in the grid and choose **Edit** from the right-click menu.

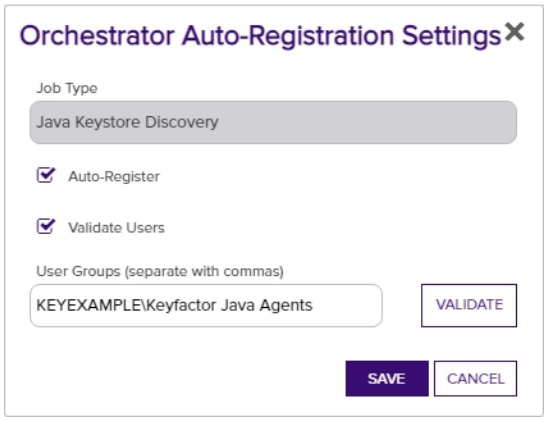


Figure 271: Orchestrator Auto-Registration Edit

3. In the Orchestrator Auto-Registration Settings dialog, check the **Auto-Register** box if you want orchestrators to be able to auto-register. If you do not enable this, an administrator will need to visit the Orchestrator Management page in the Management Portal and manually approve each orchestrator.

4. Check the **Validate Users** box if you want the users under which the orchestrators are running to be a member of a specific Active Directory group in order to auto-register. If you do not enable this but you do enable auto-registration, all orchestrators will auto-register.

    a. In the **User Groups** field, enter the AD group or groups against which to validate the user accounts in *DOMAIN\group name* format. Multiple groups should be separated by a comma and no space. User accounts may be used if desired.

    b. Click the **Validate** button to validate the entered group(s).

    Note: Validation is only supported on domain-joined Keyfactor Command servers.

5. Click **Save**.

Important: The same Active Directory group or groups in the primary Keyfactor Command forest must be used for all roles serviced by a given orchestrator type (e.g. Keyfactor Java Agent or Keyfactor Universal Orchestrator). All auto-registration settings must be populated if any are to be used even if all features are not planned for use. For example, if you plan to use, for example, Java keystores but not PEM certificate stores managed by the Keyfactor Java Agent, you still need to populate both the Java keystore and the PEM auto-registration settings to enable auto-registration for the Java Agent to function correctly. Similarly, all auto-registration settings for capabilities supported by your Keyfactor Universal Orchestrator must be populated even if you won't be using all features. Settings reserved for future use do not need to be populated, though doing so will not hurt anything.

Custom Auto-Registration Handlers

With the custom handler system of auto-registration, a handler module is written and compiled into a DLL, which is then registered in the Keyfactor Command configuration and called whenever a new orchestrator performs an initial registration request, provided there are sufficient licenses available to support the orchestrator. The handler then has the flexibility to call out to an external system such as a database or web service or use any other means to determine whether the orchestrator should be approved and what values should be applied for the blueprint, metadata, and orchestrator ClientID.

When an orchestrator first connects to Keyfactor Command, available registration handlers run in sequence to determine if the orchestrator can be automatically approved. A handler will return one of three results: Allow, Deny, and Defer. Handlers are executed in order of registration until one returns Allow or Deny or until all handlers have been executed. Whenever an executed handler returns a response of Defer, the next registered handler will be executed. If any executed handler returns a response of Deny, further processing will cease and the orchestrator will be moved into a Disapproved state. In both of these cases, values returned by the output parameters will be ignored by Keyfactor Command.

In the event of an Allow response, the following actions will occur:

- The orchestrator will be set to an Approved state.
- If the value for blueprintName corresponds to a valid orchestrator blueprint that can be applied to this orchestrator, it is applied. Otherwise, the response is rejected, the orchestrator is left with a state of New, and an error is logged.
- If the value for ClientID is non-null, it will be permanently associated with this orchestrator approval. The orchestrator will be expected to provide this value for the ClientMachine field on all future calls.
- If the CSR attribute was provided to the handler, it will be submitted for issuance and the resulting certificate will be returned to the orchestrator.
- If the request results in an issued certificate and the metadata output parameter has values, the valid metadata field values will be associated with the issued certificate.
- If ClientParameters has a value, the parameters will be returned to the orchestrator (but will not be used by Keyfactor Command).

If no handler returns a response aside from Defer, the process will continue to the built-in auto-registration system, and if the orchestrator is not approved at the conclusion of that, the orchestrator will be left in the New state for manual approval.

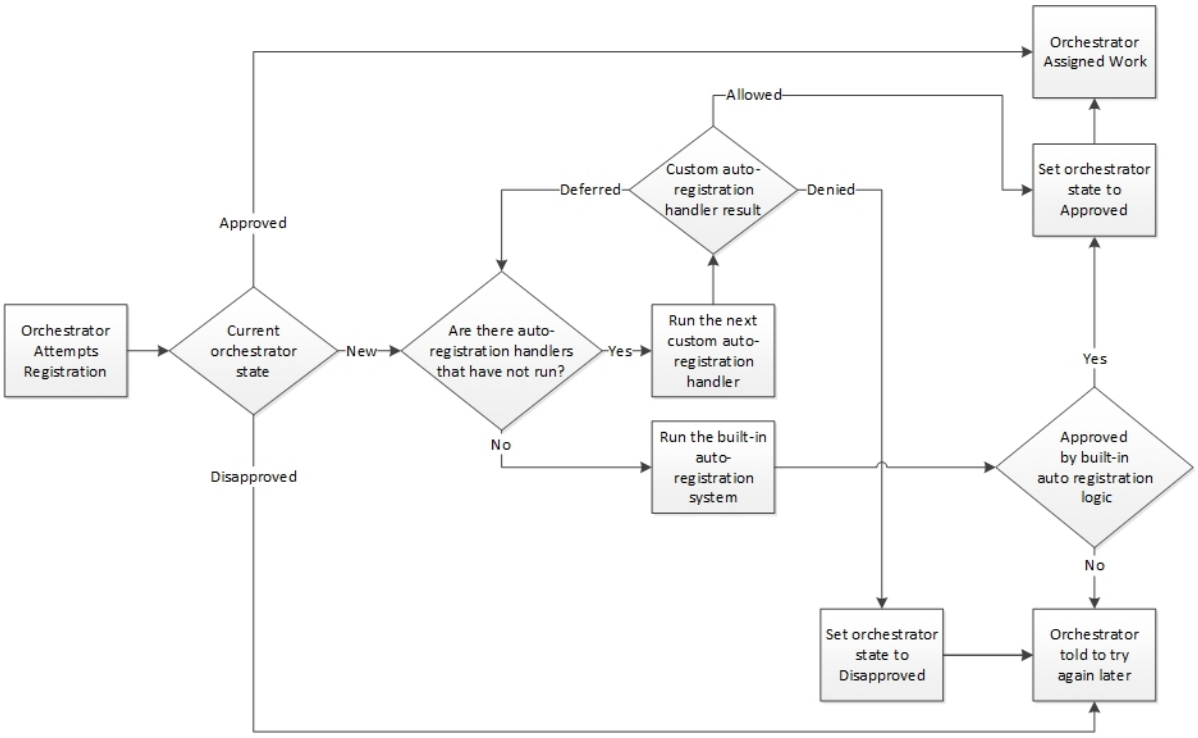


Figure 272: Orchestrator Auto-Registration Flow

Tip: Sample handler source is available as a starting point for creating a custom auto-registration handler. Contact Keyfactor support for assistance.

2.1.9.2 Orchestrator Management

Orchestrators (e.g. Keyfactor Universal Orchestrator and Keyfactor Bash Orchestrator) are managed through the Orchestrator Management page. The orchestrator management grid shows every orchestrator that is actively or has historically been in communication with the Keyfactor Command server.

The orchestrator management grid can be sorted in ascending order by clicking on a column header, with the exception of the *Capabilities* column. Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may also be adjusted by click-holding and dragging the line separating two column headers. By default, disapproved orchestrators are not included in the display. To include them, click the **Include Disapproved** box.

For a description of the columns shown in the orchestrator management grid, see Viewing Orchestrator Details on the next page.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.

You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Orchestrator Management Operations

To select a single row in the orchestrator management grid, click to highlight it and then select an operation from either the top of the grid or the right-click menu. The approve, disapprove, apply blueprint, reset, and request renewal operations can be done on multiple orchestrators at once. To select multiple rows, click the checkbox for each row on which you would like to perform an operation. Then select an operation from the top of the grid. The right-click menu supports operations on only one orchestrator at a time.

Viewing Orchestrator Details

To view details of an orchestrator, double-click the orchestrator, right-click the orchestrator and choose **View Details** from the right-click menu, or highlight the row in the grid and click **View Details** at the top of the grid. The orchestrator details dialog includes this information:

Id

The GUID of the orchestrator.

Client Machine

The host name of the orchestrator machine, either short or fully qualified depending upon how the machine reports itself.

Identity

The account the orchestrator is using to authenticate to Keyfactor Command, which may or may not be the same as the account under which the orchestrator is running. For example, the Keyfactor Universal Orchestrator service runs as a service account on the orchestrator machine that may be a local account on the machine, the Windows built-in Network Service account, or an Active Directory domain account. If it's an Active Directory account, it will be an account in the forest to which the orchestrator server is joined. The orchestrator's identity to make the connection to the Keyfactor Command server will be a service account for the identity provider configured for Keyfactor Command, which may be Active Directory, Keyfactor Identity Provider, Auth0, another OAuth 2.0 complient identity provider, or a federated identity provider. If it's Active Directory, it will be an identity from the forest in which Keyfactor Command is installed, which may be a different forest from the forest in which the orchestrator is installed.

Platform

The platform of the orchestrator—Java for the Java Agent, .NET Core for the Keyfactor Universal Orchestrator, Bash for the Keyfactor Bash Orchestrator, and ObjectiveC for the Mac agent, for example.

Version

The version number that the orchestrator has reported.

Status

Whether the orchestrator has been approved for operations with the Keyfactor Command server. Newly registered orchestrators show New in this column. Disapproved orchestrators show Disapproved.

Last Seen

The date and time when the orchestrator last contacted the Keyfactor Command server.

Capabilities

The target capabilities that are supported by that orchestrator as appropriate for the type of orchestrator. This includes custom capabilities. Some of the follow capabilities have been deprecated and will only be found on upgraded systems. For example:

- AWSCerManA
- CA
- CitrixAdc
- F5-WS-REST
- F5-SL-REST
- F5-CA-REST
- IISU
- JKS
- LOGS
- MacAutoEnrollment
- PEM
- RFJKS
- RFPkcs12
- RFPEM
- SSH
- SSL
- TemplateSync
- WinCert

Cert Store Type Short Names

The column displays a comma-separated list of the certificate stores types associated with each orchestrator. In many cases, the short name matches the capability.

Orchestrator Blueprints

The last blueprint applied to the orchestrator, if any (see Orchestrator Blueprints on page 521).

Legacy Thumbprint

The thumbprint of the certificate previously used by the orchestrator for client certificate authentication before a certificate renewal operation took place (rotating the current thumbprint into the legacy thumbprint). The legacy thumbprint is cleared once the orchestrator successfully registers with a new thumbprint.

Current Thumbprint

The thumbprint of the certificate that Keyfactor Command is expecting the orchestrator to use for client certificate authentication.

Authentication Certificate Renewal Request Status

The last error code, if any, reported from the orchestrator when trying to register a session. This code is cleared on successful session registration.

Last Thumbprint Used

The thumbprint of the certificate that the orchestrator most recently used for client certificate authentication. In most cases, this will match the *Current Thumbprint*.

Last Error Code

The last error code, if any, reported from the orchestrator when trying to register a session. This code is cleared on successful session registration.

Last Error Message

The last error code, if any, reported from the orchestrator when trying to register a session. This message is cleared on successful session registration.

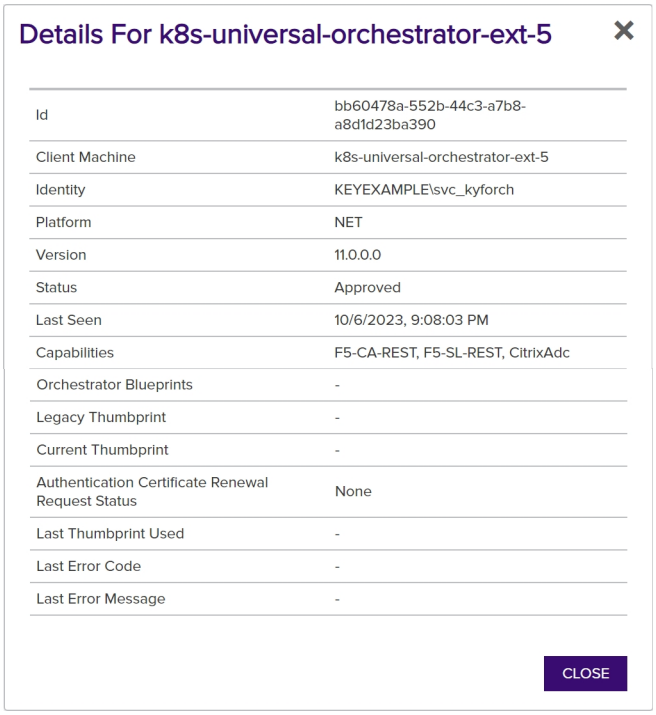


Figure 273: View Details for an Orchestrator

Approving or Disapproving Orchestrators

When orchestrators first appear in Keyfactor Command, they have a status of New. The orchestrator cannot perform any jobs while it has this status. To approve an orchestrator, highlight the row in the orchestrator management grid and click **Approve** at the top of the grid or right-click the orchestrator in the grid and choose **Approve** from the right-click menu. Once you have approved a Keyfactor Universal Orchestrator or Keyfactor Java Agent, you can schedule jobs for the orchestrator. Once you have approved an SSH Orchestrator, you can configure server groups and servers for that orchestrator and begin scanning servers. Once you have approved a Mac enroll agent, users can enroll for certificates from that Mac. Some orchestrators may be configured for auto-approval via auto-registration (see Orchestrator Auto-Registration on page 485).

To disapprove an orchestrator, highlight the row in the orchestrator management grid and click **Disapprove** at the top of the grid or right-click the orchestrator in the grid and choose **Disapprove** from the right-click menu. When an orchestrator is disapproved, operations with Keyfactor Command can no longer be carried out by this orchestrator.

Tip: An orchestrator that has been disapproved can be re-approved. Click the **Include Disapproved** check box at the top of the grid to locate the orchestrator and then approve it again. If an orchestrator that has previously been disapproved in Keyfactor Command reconnects—after an update or corrections of an issue that took it offline, for example—it will not automatically appear as New again. You will need to locate it in the disapproved orchestrators list and Approve it again.

Generating and Applying Blueprints

To generate a blueprint from an orchestrator, highlight the row in the orchestrator management grid and click **Generate Blueprint** at the top of the grid or right-click the orchestrator in the grid and choose **Generate Blueprint** from the right-click menu. For more information about blueprints, see Orchestrator Blueprints on page 521.

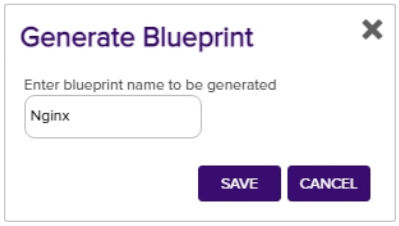


Figure 274: Generate a Blueprint from an Existing Orchestrator

To apply a blueprint to an orchestrator, highlight the row in the orchestrator management grid and click **Apply Blueprint** at the top of the grid or right-click the orchestrator in the grid and choose **Apply Blueprint** from the right-click menu. For more information about blueprints, see Orchestrator Blueprints on page 521.

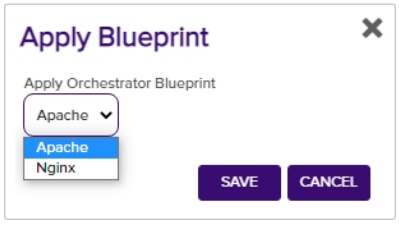


Figure 275: Apply a Blueprint from a New Orchestrator

Orchestrator Reset

The orchestrator reset and renewal functions are both useful for orchestrator maintenance. The reset function can be used when an orchestrator is in an error state or if you've made some changes on the orchestrator side that necessitate a refresh. The orchestrator reset function:

- Removes all current orchestrator jobs for the selected orchestrator.
- Deletes all associated certificate stores.
- Sets the orchestrator status to new.
- For orchestrators configured to use client certificate authentication, clears the certificate thumbprints stored for the orchestrator to allow it to be reconfigured with a new certificate.

To reset an orchestrator, highlight the row in the orchestrator management grid and click **Reset** at the top of the grid or right-click the agent in the grid and choose **Reset** from the right-click menu.

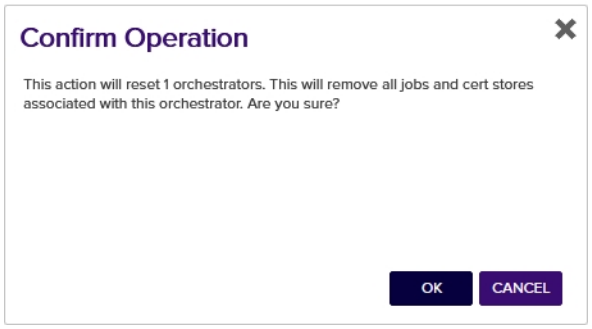


Figure 276: Reset an Orchestrator

Orchestrator Renewal

The renewal function is used for orchestrators that are authenticating via client certificate to initiate a client certificate renewal before this would occur automatically based on approaching certificate expiration. The renewal function requests or requires that the orchestrator enroll for a new client authentication certificate on the orchestrator's next session registration. It is used in conjunction with a custom renewal extension on the orchestrator to force the orchestrator to enroll for a new certificate before it would normally do so based on the warning and expiry windows. See Register a Client Certificate Renewal Extension on page 2961 for more information and custom renewal extensions on the renewal process.

To request certificate renewal for an orchestrator, highlight the row in the orchestrator management grid and click **Request Renewal** at the top of the grid or right-click the agent in the grid and choose **Request Renewal** from the right-click menu. In the Renewal Status dropdown, select one of the available options:

- None
  Unset the value so that the orchestrator will not request a new client authentication certificate (based on this value).
- Request
  The orchestrator will request a new client authentication certificate when it next registers for a session. Orchestrator activity will be allowed to continue as usual.
- Require
  The orchestrator will request a new client authentication certificate when it next registers for a session. A new session will not be granted and orchestrator activity will not be allowed to continue until the orchestrator acquires a new certificate.

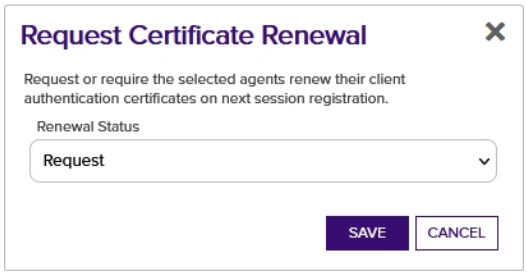


Figure 277: Request Renewal for an Orchestrator

Viewing Active Jobs for an Orchestrator

To view all the active jobs for an orchestrator, highlight the row in the orchestrator management grid and click **View Jobs** at the top of the grid or right-click the orchestrator in the grid and choose **View Jobs** from the right-click menu. This will take you to the scheduled jobs tab of the orchestrator job status page with the query field populated by the selected orchestrator.

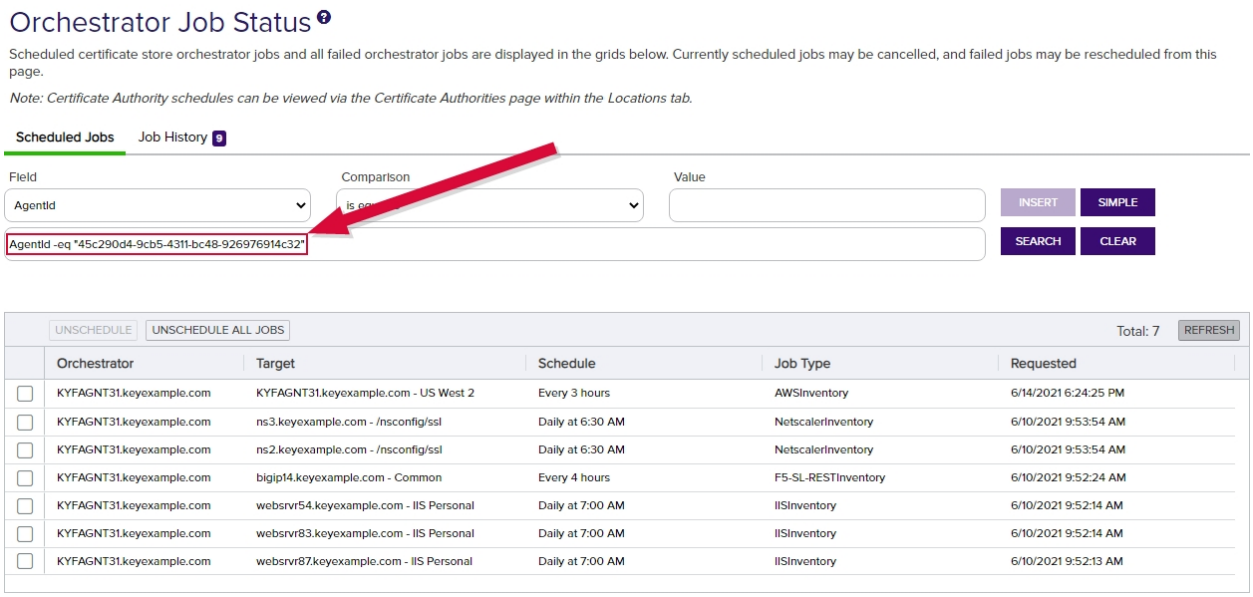


Figure 278: View Active Jobs for an Orchestrator

Viewing the Job History for an Orchestrator

To view job history for an orchestrator, highlight the row in the orchestrator management grid and click **View Job Histories** at the top of the grid or right-click the orchestrator in the grid and choose **View Job Histories** from the right-click menu. This will take you to the job history tab of the orchestrator job status page with the query field populated by the selected orchestrator.

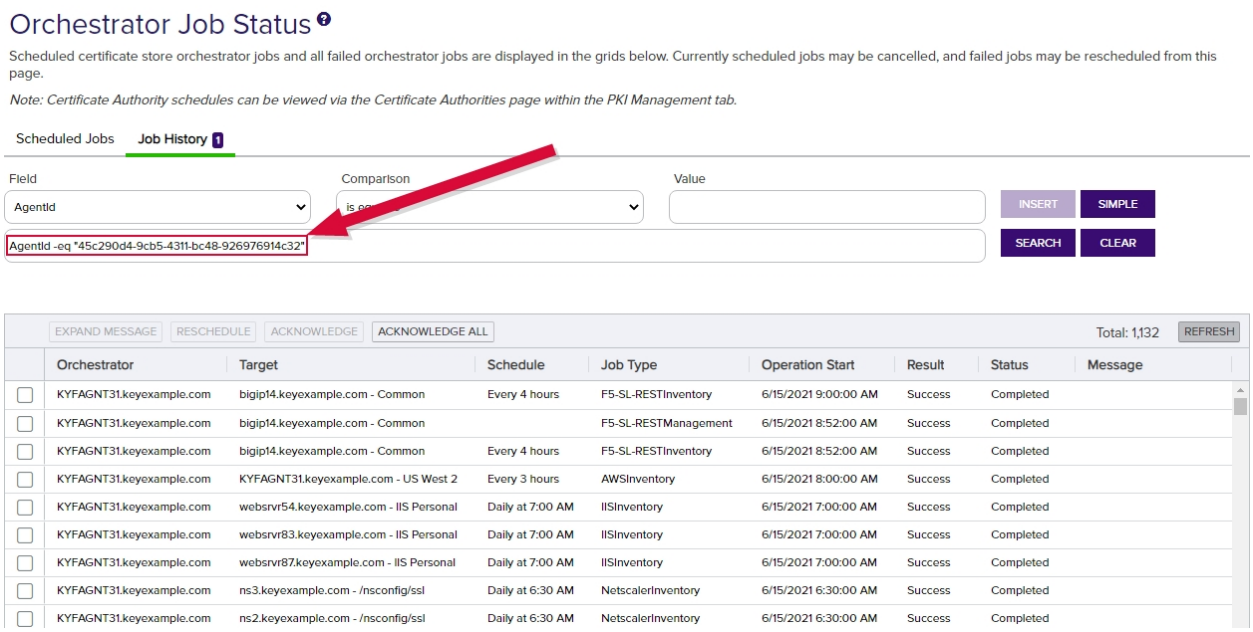


Figure 279: View Job History for an Orchestrator

Viewing Certificate Stores Associated with an Orchestrator

To view the certificate stores associated with an orchestrator, highlight the row in the orchestrator management grid and click **View Certificate Stores** at the top of the grid or right-click the orchestrator in the grid and choose **View Certificate Stores** from the right-click menu. This will take you to the certificate stores page with the query field populated by the selected orchestrator.

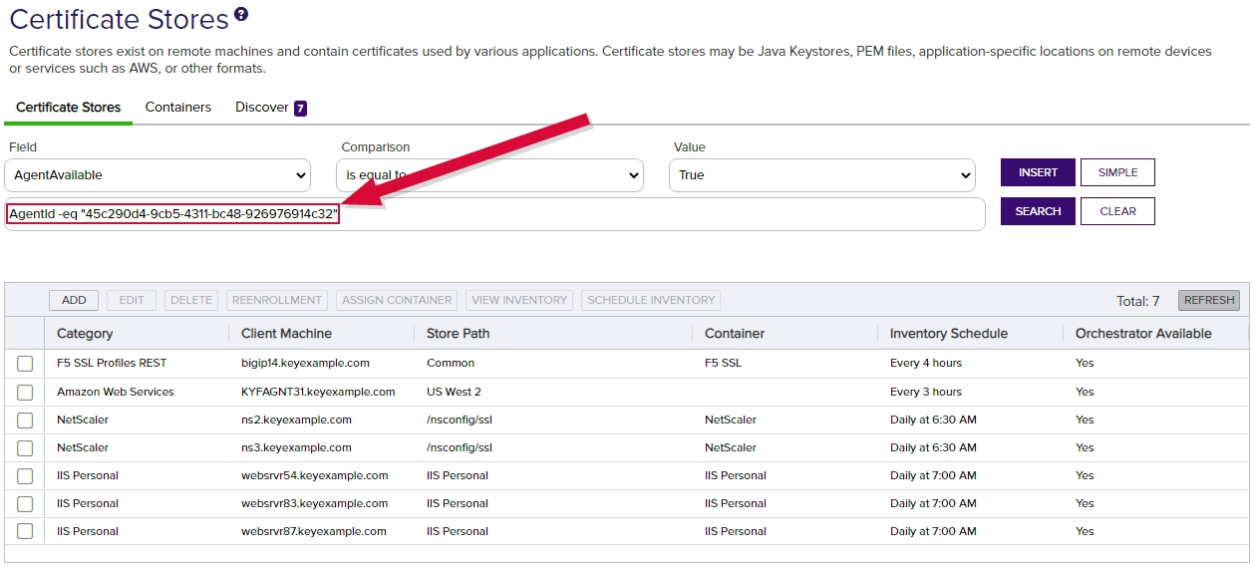


Figure 280: View Certificate Stores for an Orchestrator

Tip: This option is only useful for orchestrators that have a capability that makes use of certificate stores (e.g. JKS, PEM, IIS, etc. not SSL or SSH).

Fetch Logs

The fetch logs function is designed to retrieve a portion of the tail end of the orchestrator log for easy review. It is supported for both the Keyfactor Universal Orchestrator and the Native Agent.

To schedule a job to fetch the logs, click **Fetch Logs** from the actions buttons at the top of the Orchestrator Management grid or from the right-click menu. The job will be scheduled to run immediately, which means it should complete within a few minutes depending on other activity occurring at the same time. The fetch logs job will appear in Scheduled Jobs under Orchestrator Job Status with a job type of *Fetch Logs* and when complete will appear in Job History (see Job History on page 516).

For Native Agent fetch log jobs, when the job is complete, locate the completed job on the Job History tab and double-click or click **Expand Message** from the right-click menu or at the top of the grid. The job status message details show 4000 characters of the tail end of the log.

To review the log data for logs fetched from a Keyfactor Universal Orchestrator, use the *GET /OrchestratorJobs/JobStatus/Data* Keyfactor API method. See GET Orchestrator Jobs Job Status Data on page 1843 for more information.

Tip: The orchestrator must be approved and have the LOGS capability in order for the *Fetch Logs* function to be enabled.

Note: The orchestrator must be configured to write log entries to a file in order for the *Fetch Logs* function to be able to retrieve logs. The Keyfactor Universal Orchestrator does this by default, but the Native Agent needs to be configured appropriately to write to a file in order to support this feature.
To set up logging on the Native Agent, see the Native Agent configuration instructions to configure logging and start the orchestrator with the appropriate logging level to allow for the use of the **Fetch Logs** feature:

https://github.com/Keyfactor/Keyfactor-CAgent

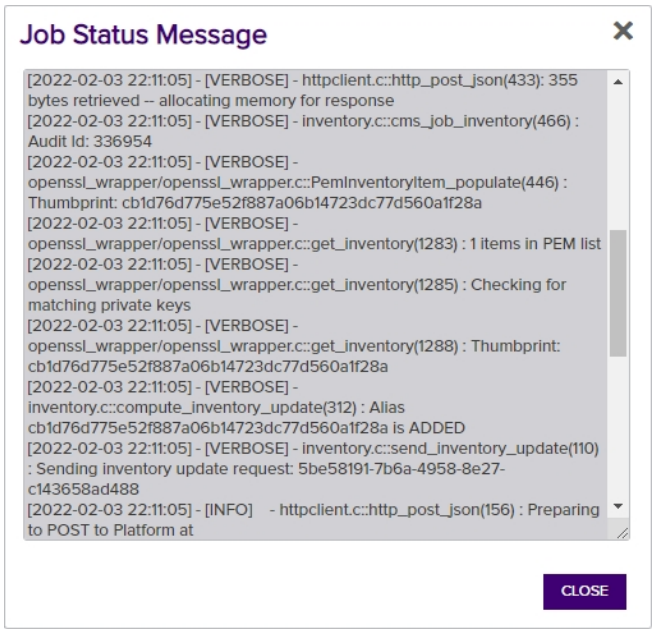


Figure 281: Sample Native Agent Fetch Log Results

Tip: If jobs for the Keyfactor Universal Orchestrator fail with messages similar to the following:

```
2021-08-05 10:47:23.1940
Keyfactor.Orchestrators.JobExecutors.OrchestratorJobExecutor [Debug] - Response
status code does not indicate success: 413 (Request Entity Too Large).

at System.Net.Http.HttpResponseMessage.EnsureSuccessStatusCode() in /_
/src/System.Net.Http/src/System/Net/Http/HttpResponseMessage.cs:line 172

at Keyfactor.Orchestrators.Services.HttpService.SendPostAsync[T](String uri,
Object requestData, Dictionary`2 headers) in F:\BuildAgents\Default1\_
work\24\s\src\OrchestratorServices\HttpService.cs:line 38
```

This indicates that the amount of data being returned on the job is greater than IIS on the Keyfactor Command server is configured to accept. You will need to make modifications to the IIS settings on your Keyfactor Command server to allow it to accept larger incoming pieces of content. You can do this using the configuration editor built into the IIS management console. Make the setting changes at the Default Web Site level (or other web site, if you installed your Keyfactor Command in an alternate web site). There are three settings that may need modification:

- system.webServer/security/requestFiltering/requestLimits/maxAllowedContentLength
- system.webServer/serverRuntime/uploadReadAheadSize

- system.web/httpRuntime/maxRequestLength

The most important of these is *maxAllowedContentLength*. Set this value to at least 2,500,000 bytes to support the maximum returned data size for the Keyfactor Universal Orchestrator. The default values of 4096 KB for the *maxRequestLength* and 49,152 for *uploadReadAheadSize* will probably be sufficient in most environments, unless you are also using SSL scanning (see Monitoring Network Scan Jobs with View Scan Details on page 465). (The system.webServer values are set in bytes while the system.web values are set in kilobytes.)

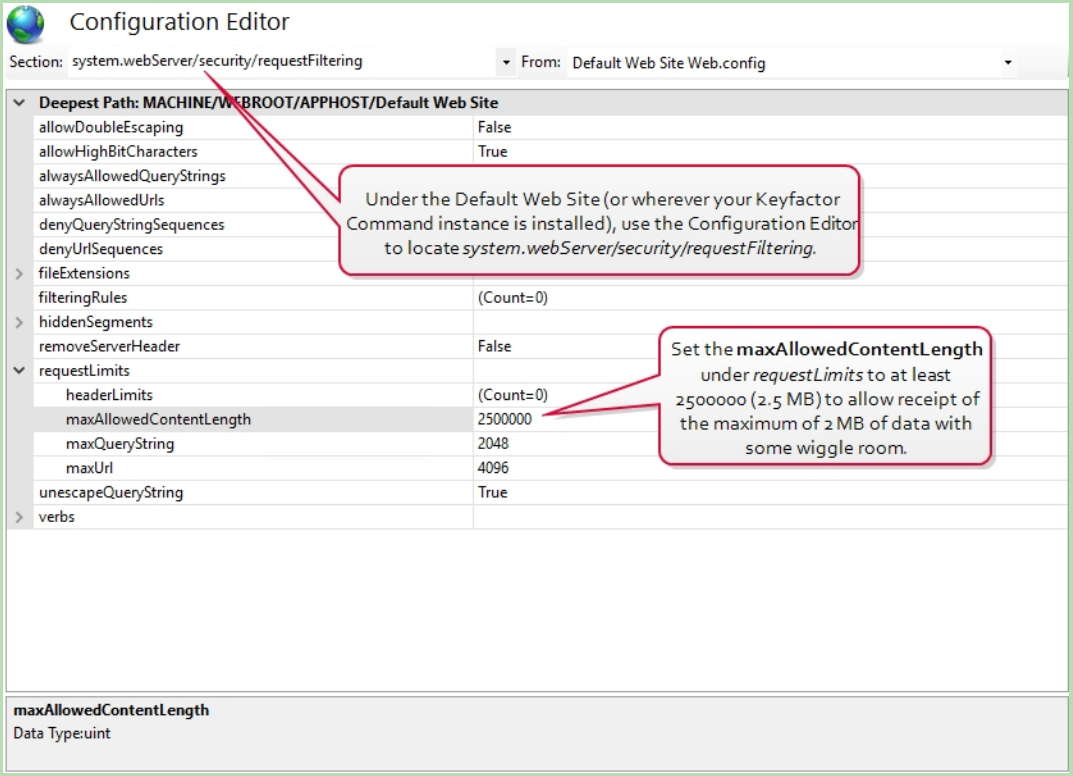


Figure 282: Modify IIS Settings for Keyfactor Universal Orchestrator Custom Jobs: maxAllowedContentLength

Using the Orchestrator Management Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

ClientMachine

Complete or partial matches with the orchestrator name as listed in the Client Machine field.

LastSeen

Orchestrator last contacted the Keyfactor Command server before, after or on a specified date.

Platform

Platform matches or doesn't match the referenced platform. Supported platforms are:

- Java (JKS and PEM)
- NET (e.g. F5, Remote File JKS, IIS, Citrix/NetScaler, Remote File PEM, Remote File PKCS#12, and SSL)
- Mac
- Android
- Native
- Bash (SSH)
- Unknown

Status

Status matches the selected category—New, Approved or Disapproved.

Identity

Complete or partial matches with the account the orchestrator used when registering with the Keyfactor Command server.

Capabilities

Capability matches the referenced capability. Supported capabilities, including common Keyfactor custom-built extensions with default capability names, include (plus any custom types you've created):

- AWSCerManA
- CA
- CitrixAdc
- F5-WS-REST
- F5-SL-REST
- F5-CA-REST
- IISU
- JKS
- LOGS
- MacAutoEnrollment
- PEM
- RFJKS
- RFPkcs12
- RFPEM
- SSH
- SSL
- TemplateSync
- WinCert

Version

Complete or partial matches with the version the orchestrator reported when registering with the Keyfactor Command server.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and

then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

Important: The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

2.1.9.3 Orchestrator Job Status

The Orchestrator Job Status page provides information on currently scheduled certificate store, SSH, and SSL jobs as well as an audit log of job history.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Scheduled Jobs

The Scheduled Jobs tab on the Orchestrator Job Status page shows all of the currently scheduled jobs for any approved Android, Java, Native, and SSH Orchestrators and jobs other than remote CA sync for approved Keyfactor Universal Orchestrators (SSL jobs only appear while they are in progress). At a glance, you can see what discovery, inventory, management, and synchronization jobs are scheduled for all the active orchestrators that can communicate with Keyfactor Command.

The Orchestrator Job Status grid includes these fields:

Orchestrator

The host on which the orchestrator is running.

Target

The target machine name followed by the path and file name to the certificate store on the target machine for many types of jobs. This field may be blank for some types of jobs.

Schedule

The time at which or frequency with which a job will run. Add and remove certificate jobs will show *Immediately* unless they have been scheduled for a later time. Renewals and reenrollments will always show *Immediately* since these can't be scheduled for a later time. SSL jobs will always show *Immediately* since they only appear in the grid while they are in progress.

Job Type

The type of job—e.g. inventory, discovery, management (add and remove certificate), synchronization.

Requested

The date and time when the job was configured or updated.

The operations available on the Scheduled jobs tab are:

Unschedule

To unschedule a job, highlight the row for the job in the orchestrator job status grid and click **Unschedule** at the top of the grid or right-click the job in the grid and choose **Unschedule** from the right-click menu.

Unschedule All Jobs

To unschedule multiple jobs, do a search for the jobs you wish to unschedule (e.g. JobType -contains "Discovery") and click **Unschedule All Jobs** at the top of the grid.

If an inventory job for a certificate store is unscheduled, all instances of that job will be removed (as opposed to just the next inventory job) and that store will not be inventoried again until another inventory job is scheduled for it on the Certificate Stores page.

Tip: SSL discovery and monitoring jobs and SSH synchronization jobs cannot be unscheduled from this page—this should be done in SSL and SSH management instead (see SSL Discovery on page 453 and SSH Server Groups on page 560).

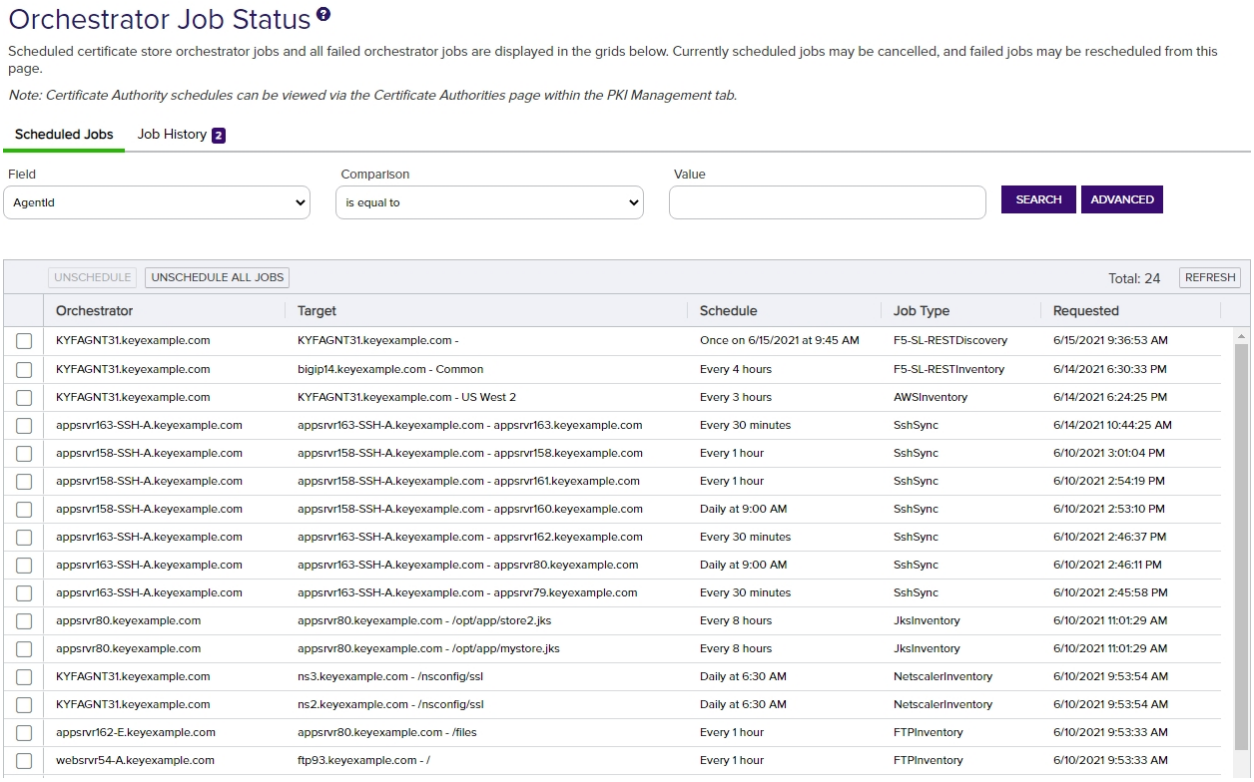


Figure 283: Orchestrator Job Status Scheduled Jobs

Scheduled Job Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

AgentMachine

Complete or partial matches with the orchestrator name as listed in the Orchestrator Machine field.

TargetPath

Complete or partial matches with the contents of the Target field, including the target machine name and the certificate store path and file name.

ScheduleType

Schedule Type matches or does not match the referenced type. Supported schedule types are:

- Immediate
- Interval
- Daily
- Weekly
- Monthly
- Once

Requested

Job was requested or updated before, after or on a specified date. Supports the %TODAY% token (see Advanced Searches on page 515).

JobType

Job Type contains or doesn't contain the referenced keywords. Supported keywords are:

- Management (including add and remove certificates)
- Inventory
- Certstore Discovery
- SSL Discovery
- Reenrollment
- SSL Monitoring
- Sync
- Enrollment

AgentType

Orchestrator Type matches the referenced type. Supported orchestrator types , including common Keyfactor custom-built extensions with default names, include (plus any custom types you've created):

- AWSCerManA
- CA
- CitrixAdc
- F5-WS-REST
- F5-SL-REST
- F5-CA-REST
- IISU
- JKS
- LOGS
- MacAutoEnrollment
- PEM
- RFJKS
- RFPkcs12
- RFPEM
- SSH
- SSL
- TemplateSync

- WinCert

AgentPlatform

Orchestrator Platform matches or doesn't match the referenced platform. Supported platforms are:

- Java (JKS and PEM)
- NET (e.g. F5, Remote File JKS, IIS, Citrix/NetScaler, Remote File PEM, Remote File PKCS#12, and SSL)
- Mac
- Android
- Native
- Bash (SSH)
- Unknown

AgentID

Orchestrator ID matches or doesn't match the entered GUID (primarily used for internally generated searches when the user is redirected here from another page).

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

In addition to the options available in the query builder, three special values can be used in selected searches by typing them in directly:

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a

domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).

- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

> **Important:** The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

Job History

The Job History tab on the Orchestrator Jobs page shows a record of discovery, inventory and management jobs for certificate stores, SSH servers, SSL endpoints and remote CAs. It keeps the three most recent inventory jobs, whether they have warnings, failed, or succeeded. Information on potential causes of the problem to allow for troubleshooting is provided for failed jobs.

There are several settings that control the behavior of orchestrator jobs in *Settings > Application Settings-Agent Tab* (see Application Settings: Agents Tab on page 614), most notably:

- *Job Failures and Warnings Age Out (days)*
- *Notification Alert Email Recipients*
- *Notification Alert Interval (minutes)*
- *Number of times a job will retry before reporting failure*
- *Number of times a job will retry connecting to PAM providers*

The operations available on the Job History tab are:

Expand Message

To view the details of an error or warning message, double-click the row for the job in the orchestrator job history grid, right-click the job and choose **Expand Message** from the right-click menu, or highlight the row in the grid and click **Expand Message** at the top of the grid.

Reschedule

To reschedule a job, correct the error that caused the problem, then highlight the row for the job in the orchestrator job history grid and click **Reschedule** at the top of the grid or right-click the job in the grid and choose **Reschedule** from the right-click menu.

Acknowledge [All]

To mark an error or warning grid entry as acknowledged, highlight the row for the job in the orchestrator job history grid and click **Acknowledge** at the top of the grid or right-click the job in the grid and choose **Acknowledge** from the right-click menu. Jobs that are in process or that have completed successfully cannot be marked as acknowledged. Marking a job as acknowledged

removes it from the count on the job history tab (if the job falls within the count period defined by the *Job Failures and Warnings Age Out (days)* application setting—see Application Settings: Agents Tab on page 614).

The small number that appears on the tab to the right of the title indicates how many failures and warnings there have been, if any, within the last seven days, by default, unless the job has been marked as acknowledged. This acts as a reminder to check for failures and warnings. This number of days for reporting is configurable using the *Job Failures and Warnings Age Out (days)* application setting (see Application Settings: Agents Tab on page 614).

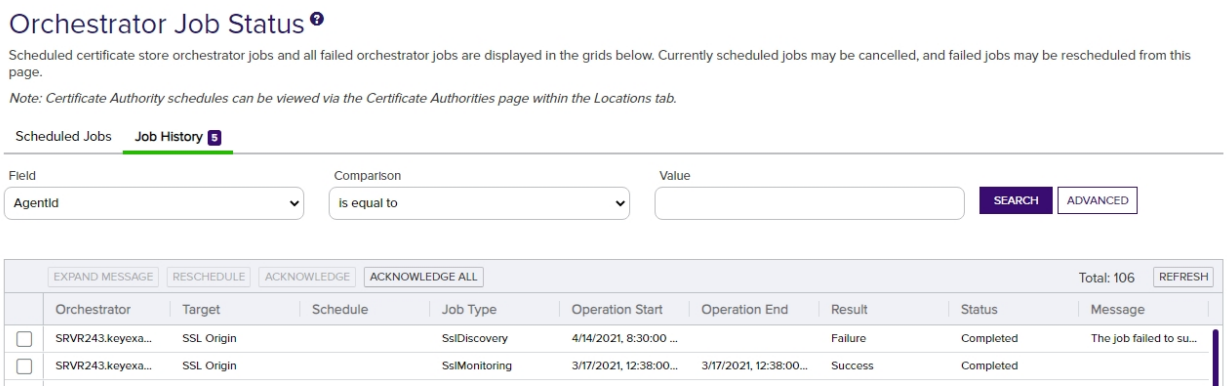


Figure 284: Orchestrator Job History

The Job History grid includes these fields:

Orchestrator

The host on which the orchestrator was running. [All]

Target

The target machine name followed by the path and file name to the certificate store on the target machine for many types of jobs, the endpoint group name for SSL jobs, or the CA name for remote CA jobs. This field may be blank for some types of jobs.

Schedule

The time at which or frequency with which a job was scheduled to run. Add and remove certificate jobs, will show *Immediately* unless they were scheduled for a later time. Renewal and reenrollment jobs will always show *Immediately* since they don't support later scheduling, as will fetch logs jobs.

Job Type

The type of job that was run and in some cases the orchestrator type associated with the job (e.g. F5 SSL Profiles Management, PEM File Discovery, Java Keystore Inventory or CA Synchronization).

Operation Start

The time at which the job was run.

Operation End

The time at which the job was completed.

Result

The outcome of the job—e.g. Success, Failure, or Warning. Under some circumstances—for example, jobs that are still actively running—Unknown may appear here.

Status

The status of the job—e.g. Acknowledged, Completed, CompletedWillRetry, or InProcess. If a job shows as CompletedWillRetry, it has failed at least once, is automatically retrying five times, by default (see the *Number of times a job will retry before reporting failure* in Application Settings: Agents Tab on page 614) and cannot be rescheduled because it is still attempting to run.

Message

The message indicating the reason for the failure or warning, if applicable. Double-click the grid row or right-click and choose **Expand Message** from the right-click menu to read the error message in full.

Note: Currently, any jobs initiated with the **Fetch Logs** function will not be included in any **Job Type** search results, but will be included in any other query search field. See Fetch Logs on page 505 for more information.

Note: For Bash Orchestrator message resolution see SSH-Bash Orchestrator Job History Warning Resolution on page 790.

Job History Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

Status

Status matches or doesn't match the selected category—Acknowledged, Completed, InProcess, Waiting, Unknown.

Result

Result matches or doesn't match the selected category—Failure, Warning, Success, Unknown.

Agent

Complete or partial matches with the orchestrator name as listed in the orchestrator field.

TargetPath

Complete or partial matches with the contents of the Target field, including the target machine name and the certificate store path and file name for types of jobs listing those, or for SSL jobs, the endpoint group name, or for remote CA synchronization jobs, the CA name.

ScheduleType

Schedule Type matches or does not match the referenced type. Supported schedule types are:

- Immediate
- Interval
- Daily
- Weekly
- Monthly
- Once

JobType

Job Type matches or does not match the referenced type. Supported categories are:

- Management
- Inventory
- Certstore Discovery
- SSL Discovery
- Reenrollment
- SSL Monitoring
- CA Synchronization

OperationStart

Operation Start before or after a specified date and time. Supports the %TODAY% token (see Advanced Searches on the next page).

Message

Partial matches with the error or warning message listed in the Message field.

AgentID

Agent ID matches or doesn't match the entered GUID (primarily used for internally generated searches when the user is redirected here from another page).

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

In addition to the options available in the query builder, three special values can be used in selected searches by typing them in directly:

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

> **Important:** The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

2.1.9.4 Orchestrator Blueprints

The orchestrator blueprint system allows a large number of similar orchestrators to be configured with minimal effort on the part of the user. By taking a snapshot of the certificate stores and scheduled jobs on one orchestrator, matching certificate stores and jobs can be defined on another orchestrator with just a few clicks. With an orchestrator auto-registration handler, blueprint application can even be completely automated, so that a large number of machines or devices can be configured and obtain certificates with no user input after initial configuration of the blueprint and handler. This can greatly improve security by ensuring that each device is provisioned from day one with a unique certificate using a private key generated on the device as well as an up-to-date list of trusted roots, and it allows for continuous monitoring and reporting of all certificates across all configured devices.

Orchestrator blueprints are generated from the Orchestrator Management page (see Orchestrator Management on page 496) and applied to new orchestrators manually via the Orchestrator Management page. On the Orchestrator Blueprints page, you can review the existing blueprints, view details of a blueprint (what certificate stores and scheduled jobs are included in the blueprint), and delete blueprints.

Blueprint Operations

Some blueprint operations are carried out on the Orchestrator Management page (generating and applying blueprints) while others are done on the Orchestrator Blueprints page (viewing and deleting blueprints).

Applying Blueprints

When you apply a blueprint to an orchestrator, you are defining a set of certificate stores and scheduled jobs for that orchestrator as determined by the blueprint at the time that the blueprint is applied. There is no ongoing effect to having a blueprint applied. If the blueprint is deleted, this does not affect the orchestrators to which the blueprint was applied. Likewise, changing the orchestrator from which the blueprint was created after creation of the blueprint does not affect the blueprint. The blueprint continues to contain the certificate stores and scheduled jobs that were associated with the orchestrator at the time the blueprint was taken.

Orchestrator blueprints work with Java and PEM certificate stores and can be used with the Java, Native, and Android agents.

Blueprints are applied to an orchestrator from the Orchestrator Management page (see Generating and Applying Blueprints on page 501).

Modifying Blueprints

Blueprints can't be edited. To modify a blueprint, modify the certificate stores and scheduled jobs on the orchestrator from which the blueprint was taken and capture a new blueprint (see Generating and Applying Blueprints on page 501). This will replace the existing blueprint. An orchestrator can only have one blueprint at a time.

Figure 285: Orchestrator Blueprints

Deleting Blueprints

To delete a blueprint:

1. In the Management Portal, browse to *Orchestrators > Orchestrator Blueprints*.
2. On the Orchestrator Blueprints page, select an orchestrator blueprint and click **Delete** from either the top or right-click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

Viewing Blueprint Details

To view the details of a blueprint:

1. In the Management Portal, browse to *Orchestrators > Orchestrator Blueprints*.
2. On the Orchestrator Blueprints page, select an orchestrator blueprint and double-click or click **View** from either the top or right-click menu.
3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

On the Certificate Stores tab you can see the certificate store paths and types that have been associated with the blueprint. On the Scheduled Jobs tab you can see the scheduled jobs for these certificate stores. These would generally be inventory jobs, though it is possible to blueprint an orchestrator with other types of active jobs (e.g. discovery).

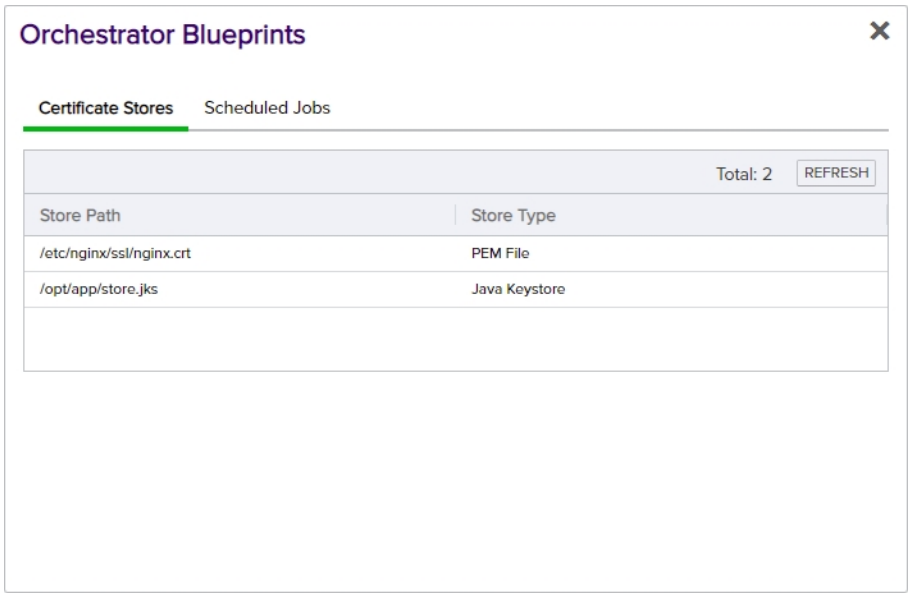


Figure 286: Orchestrator Blueprint Details: Certificate Stores Tab

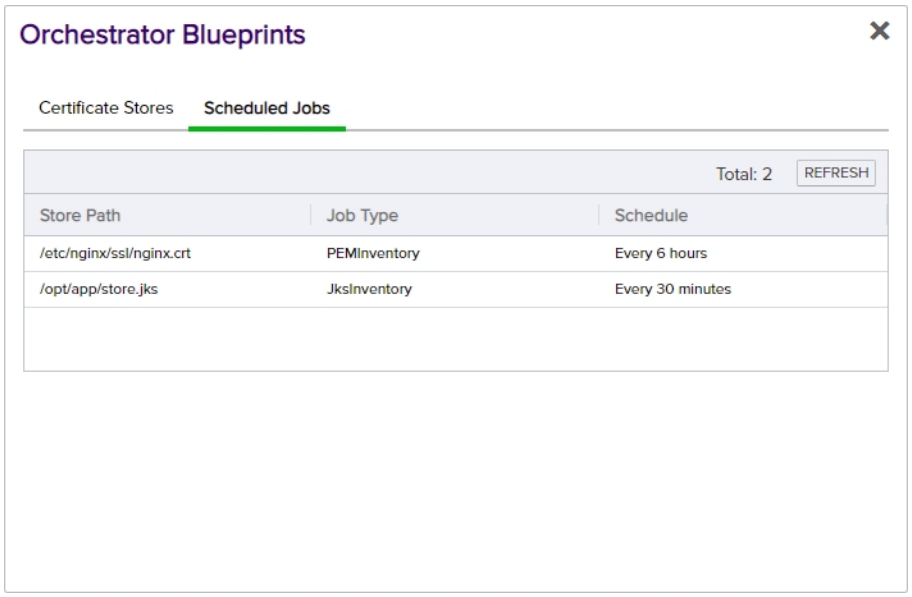


Figure 287: Orchestrator Blueprint Details: Scheduled Jobs Tab

2.1.9.5 Mac Auto-Enrollment

The settings on the Mac Auto-Enrollment page control how Mac auto-enrollment agents in your environment auto-enroll for certificates through Keyfactor Command. The available settings are:

Enabled

Controls whether Mac auto-enrollment is allowed in the environment.

Interval

Defines, in minutes, how frequently the agent should check to see if there are new certificates for which to enroll.

Use Metadata

If enabled, allows you to automatically associate data in a custom metadata field with an auto-enrolled Mac certificate. See Certificate Metadata on page 710 for more information about metadata fields.

Metadata Field Name

Choose an existing metadata string or Boolean field in the dropdown to populate for the certificate, if *Use Metadata* is enabled.

Metadata Value Type

Determines whether the data inserted in the metadata field will be based on the machine from which the certificate is requested or will be set to the same value for all certificates. Choose *Special Text* to pick from machine-specific values in the Metadata Value dropdown. Choose *Static Value* to enter text that will be populated in every Mac auto-enrollment certificate that is issued.

Metadata Value

If you select Special Text for the Metadata Value Type, this field will be a dropdown including values that are available from the Mac client. In the current

version of the agent, only the Mac serial number is available. If you select Static Value for the Metadata Value Type, this will be a free-form field in which you can type any text you want to appear in the selected metadata field for all Mac auto-enrolled certificates. If you've selected a Boolean metadata field, you'll have the choice of *True* or *False* for the value.

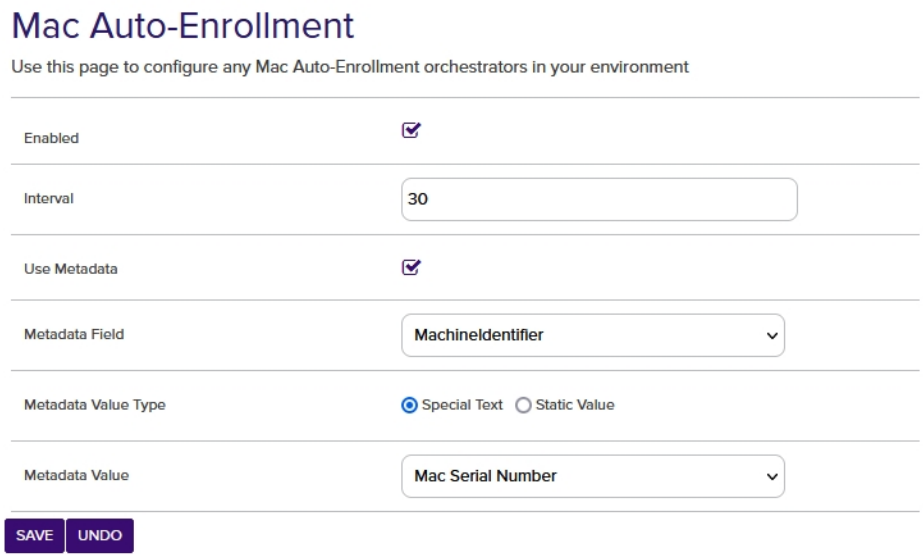


Figure 288: Mac Auto-Enrollment Configuration

To save your changes, click **Save** at the bottom of the page, or to revert to the previous settings without saving, click Undo.

Tip: For more information about the Mac Auto-Enrollment Agent, see the separate Mac Auto-Enrollment Guide.

2.1.10 SSH

Keyfactor SSH Management is designed to allow organizations to inventory and manage secure shell (SSH) keys across the enterprise. The solution consists of two elements; the SSH functionality on the Keyfactor Command Management Portal and the Keyfactor Bash Orchestrator.

Important: SSH management in Keyfactor Command with the Keyfactor Bash Orchestrator is only supported when using Active Directory as an identity provider (see Selecting an Identity Provider for Keyfactor Command on page 2704). The SSH option in the Management Portal will only appear when Keyfactor Command is installed using Active Directory as an identity provider (and with a license that supports SSH).

The Keyfactor Bash Orchestrator runs on Linux servers and can be operated in two possible modes:

- The orchestrator is used in *inventory only* mode to perform discovery of SSH public keys and associated Linux user accounts across multiple configured targets.

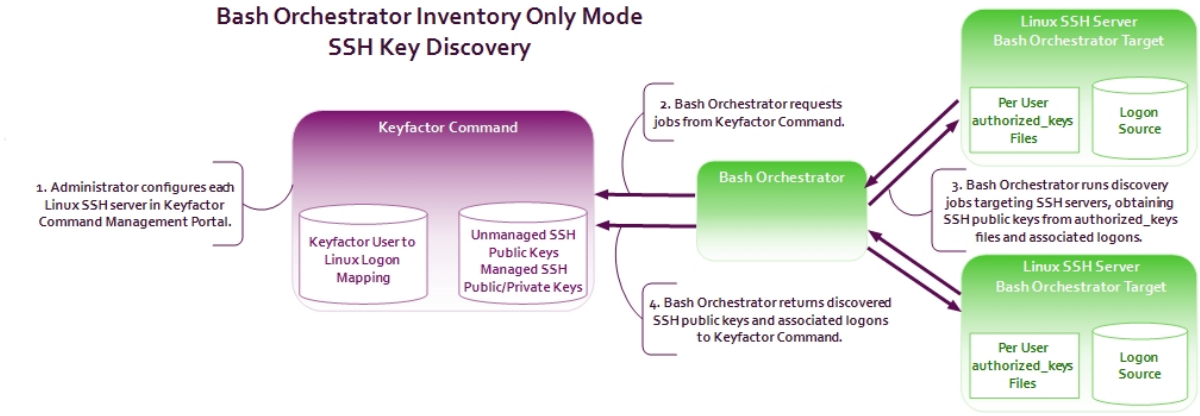


Figure 289: SSH Key Discovery Flow

- When operated in *inventory and publish policy* mode, the orchestrator can be used to add SSH public keys and Linux user accounts on targets and remove rogue keys that appear without authorization.

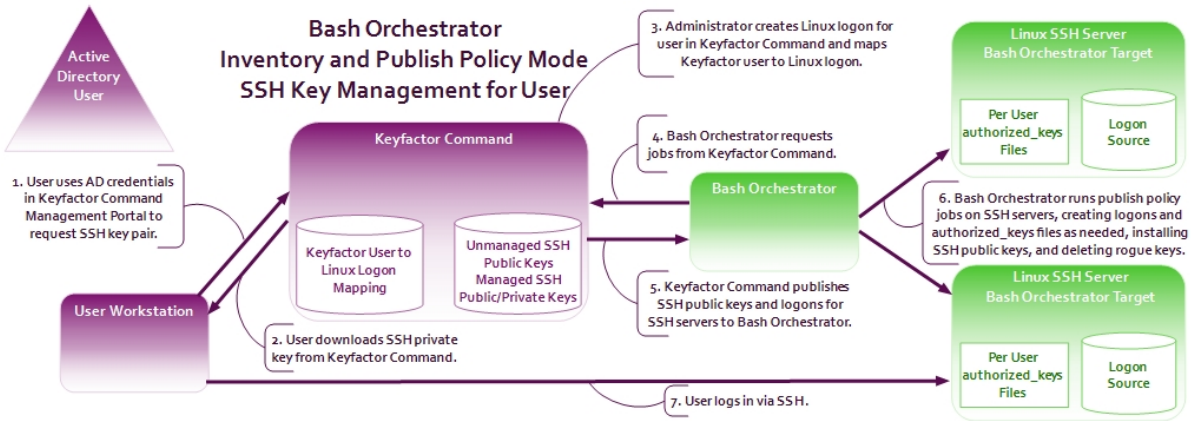


Figure 290: SSH User Key Management Flow

As you work with SSH keys in Keyfactor Command, you will need to understand the difference between *users*, *service accounts*, and *logons*:

- A *user* is an account in Keyfactor Command—based on an Active Directory user account—which has been granted the Keyfactor Command SSH User role permission (see SSH Permissions on page 597).

  A *user* can use the My SSH Key tool (see My SSH Key on page 531) to generate an SSH key pair for himself or herself. This stores the user's SSH public and private key in the Keyfactor Command database. An administrator can then use one of the options in the SSH section of the Management Portal (see Editing Access to an SSH Server on page 579, Editing Access to an SSH Server Group on page 563, or Adding Logons on page 585) to map the *user* record and its associated public key to one or more *logons*, creating new *logons* if needed. For servers

operating in *inventory and publish policy* mode, this will cause the *user*'s public key to be published to the authorized_keys file(s) for each mapped *logon* on the associated SSH server(s) during the next synchronization job. The *user* downloads the private key of the key pair to his or her machine in the My SSH Key tool and retains it there to allow for SSH connections to the target servers the administrator distributes the matching public key to.

Note: If an administrator maps a *user*'s public key to a *logon* for a server that is in *inventory only* mode, nothing will happen. The key will not be published to the server.

Note: OpenSSH maintains a file for each user that contains the public keys authorized to connect via SSH. By default, this file is named authorized_keys. In this document, we refer to this file as *authorized_keys*, however in your environment, this file may have a different name. The file name used in a given environment is defined in the AuthorizedKeysFile setting in the OpenSSH sshd_config file.

- A *service account* is a string representing a service for which an SSH key has been requested through the Service Account Keys page (see Service Account Keys on page 542). It is made up of the *Username* and *Client Hostname* entered during service account key creation in the form servicename@hostname (e.g. myservice@appsrvr12).

  Tip: The client hostname that makes up part of the service account name is not necessarily an actual server hostname. It is a user-defined reference that can contain any string.

  An administrator can use the Service Account Keys page (see Service Account Keys on page 542) to generate an SSH key pair for an application—referenced by a *service account* name—that makes use of SSH for communication, storing the application's SSH public and private key in the Keyfactor Command database. The administrator needs to store the private key securely on the Linux server where the service account for the application can access it and follow the same procedure as for users to distribute the public key to the appropriate SSH server(s) operating in *inventory and publish policy* mode.

  Note: If an administrator maps a *service account*'s public key to a *logon* for a server that is in *inventory only* mode, nothing will happen. The key will not be published to the server.

- A *logon* is a Linux user account. In most cases for the purposes of SSH management, these are Linux user accounts that have or are intended to have SSH public keys associated with them on managed SSH servers, stored in an authorized_keys files. However, Linux *logons* without keys (and which should likely never have keys like "root" or OS-specific accounts like "halt") also appear in Keyfactor CommandSSH management.

Typically, you would initially configure your servers in *inventory only* mode and scan the servers for any existing authorized_keys files containing SSH public keys. This is the discovery phase. Once the discovery phase is complete for a server or server group, you would then switch it to *inventory and publish policy* mode.

When a server is in *inventory and publish policy* mode, any new keys that appear in its authorized_keys files in a manner other than by distribution from Keyfactor Command are automatically deleted. This allows administrators to closely control who has access to the servers via SSH. Any keys and authorized_keys files that were in place before the switch to managed mode are synchronized to Keyfactor Command (see Unmanaged SSH Keys on page 556) but not removed from the Linux server. The administrator can choose to remove them through Keyfactor CommandSSH management once the switch to *inventory and publish policy* mode is made, if desired. Any keys placed on the Linux server via Keyfactor Command once the servers are in *inventory and publish policy* mode are considered managed keys and do not appear on the Unmanaged Keys page.

As SSH servers are scanned for SSH keys during the initial discovery phase, the Linux user accounts associated with these keys are synchronized to Keyfactor Command. These user accounts—logons—can be viewed on the Logons tab under Server Manager. Once each server is switched to *inventory and publish policy* mode, these logons can be managed and additional logons can be added to the Linux servers via Keyfactor CommandSSH management.

Example: A large organization has dozens of Linux servers that have historically been accessed using SSH public key authentication. They don't know who has access to which servers using this method or what public keys are out on the servers. To get the keys under control, they first do discovery:

1. Install the Keyfactor Bash Orchestrator on one Linux server in the environment.
2. Copy the remoteinstall.sh script, containing the public key of the orchestrator service account, from the orchestrator to the first ten Linux targets they want to bring under control.
3. On each of the control targets, run the remoteinstall.sh script. This creates a local user account and installs the orchestrator's SSH public key to allow the orchestrator to use SSH to remote into the control target to run inventory and publish policy.
4. In the Keyfactor Command Management Portal, approve the new orchestrator (see Approving or Disapproving Orchestrators on page 500).
5. In the Management Portal, create at least one server group, setting a scanning schedule of every hour (Interval = 1 hour) for the initial discovery phase and leaving the **Enforce Publish Policy** box unchecked (see Adding Server Groups on page 561).

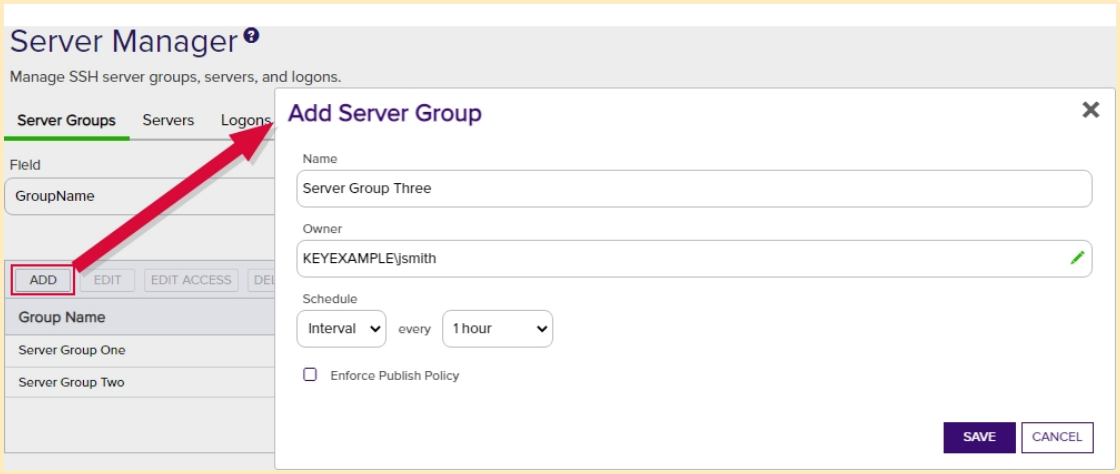


Figure 291: Add SSH Server Group for Discovery

6. In the Management Portal, add one server record for the orchestrator and one for each control target (a total of 11 records added), making them members of the group created in the previous step and selecting the **Inventory Only** radio button on the Basic tab (see Adding SSH Servers on page 577).

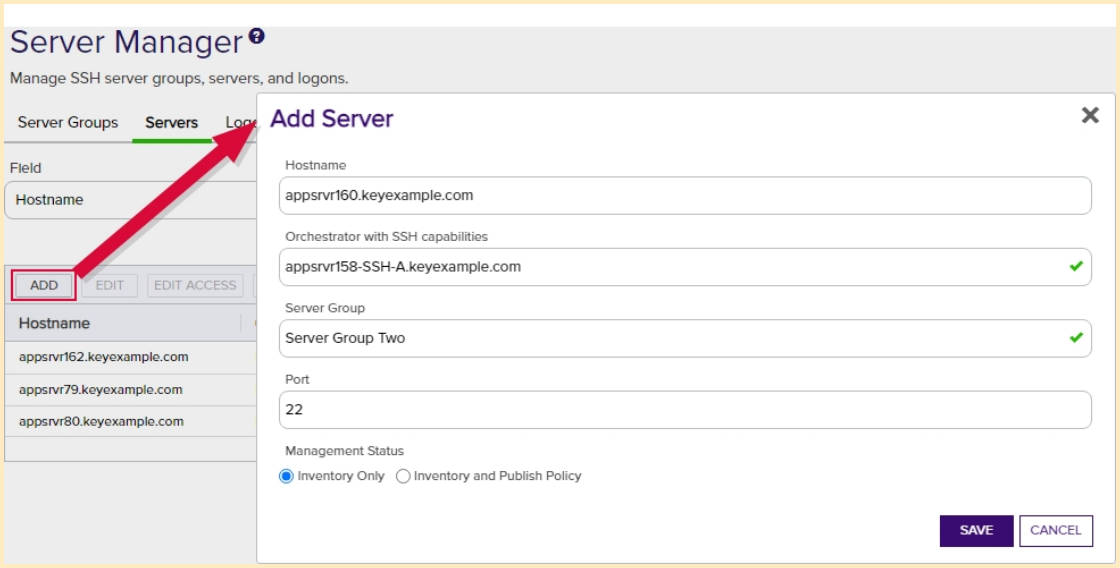


Figure 292: Add SSH Server for Discovery

7. After allowing the discovery scans to run, review the logons (see Logons on page 585) and the keys discovered (see Unmanaged SSH Keys on page 556) to see what keys are out on the servers and who they belong to.

Now having a handle on what keys are on these ten target servers plus the orchestrator itself, they are now ready to bring these servers under management. To bring the servers under management, they:

1. In the Management Portal, edit the record for the server group and check the **Enforce Publish Policy** box (see Editing or Deleting an SSH Server on page 579). This change will replicate to all servers in the group.
2. In the Management Portal, use the Logons page to remove any Linux user accounts that should not be on the target servers (see Editing or Deleting a Logon on page 588).
3. In the Management Portal, use the Unmanaged SSH Keys page to remove any public keys that are no longer needed from the target servers (see Deleting an Unmanaged Key on page 557).

These servers are now ready for ongoing management. The administrator is now ready to do discovery on the next group of servers, for which a second server group should be created.

See further examples in My SSH Key on the next page and Service Account Keys on page 542.

For more information about the orchestrator, see Bash Orchestrator on page 2991 in the *Keyfactor Orchestrators Installation and Configuration Guide*.

The options available in the SSH section of the Management Portal are:

My SSH Key

Generate an SSH key pair for the logged on user and download the private key to the local machine. The public key is stored in Keyfactor Command and can be pushed out to Linux client controlled by the Keyfactor Bash Orchestrator to allow the user access to the servers.

Service Account Keys

Generate an SSH key pair for a service using SSH and download the private key to the local machine. The public key is stored in Keyfactor Command and can be pushed out to Linux servers controlled by the Keyfactor Bash Orchestrator to allow the user access to the servers.

Unmanaged Keys

Review public SSH keys found during discovery on servers configured to be inventoried by the Keyfactor Bash Orchestrator in *inventory only* mode.

Server Manager

Manage servers, server groups, server logons for Linux clients, and SSH users controlled by the Keyfactor Bash Orchestrator.

2.1.10.1 My SSH Key

On the My SSH Key page, any user with the *SSH User* Keyfactor Command role permission (see SSH Permissions on page 597) can generate an SSH key pair for himself or herself. If the user has previously generated a key pair through Keyfactor Command, it will be displayed here. In this interface a user can view only his or her own key pair; keys for any other Keyfactor Command users are not accessible.

Example: An administrator wants to provision new user Zed Adams and grant him access to login via secured SSH using PuTTY to three Linux servers controlled by the Keyfactor Bash Orchestrator. The servers are set to both inventory and publish policy. To accomplish this, the administrator:

1. Adds Zed's AD account to the AD group that grants him the SSH User role permission in Keyfactor Command and allows him to login to the Management Portal.
2. Directs Zed to login to the Management Portal, go to the My SSH Key page and generate a new key pair (see Generating a New Key on page 536). She instructs him to enter the following information in the form:
    - **Key Type**: Ed25519
    - **Key Length**: 256
    - **Username**: Accept the default (his AD username)
    - **Email**: zed.adams@keyexample.com
    - **Passphrase**: A password of Zed's choosing used to secure the private key on download.
    - **Comment**: Zed B. Adams
3. Instructs Zed to download the SSH private key and use the PuTTY Key Generator tool to open the key and convert it to the PuTTY format:
    a. Click **Load** and browse to locate the downloaded private key. This key is named something like *SSH-Key-KEYEXAMPLE-zadams.identity*.
    b. In the Parameters section of the page, select **Ed25519** as the type of key to generate.
    c. Click **Save private key** and save the private key in the PuTTY format (*.ppk) in a safe location on the local machine.

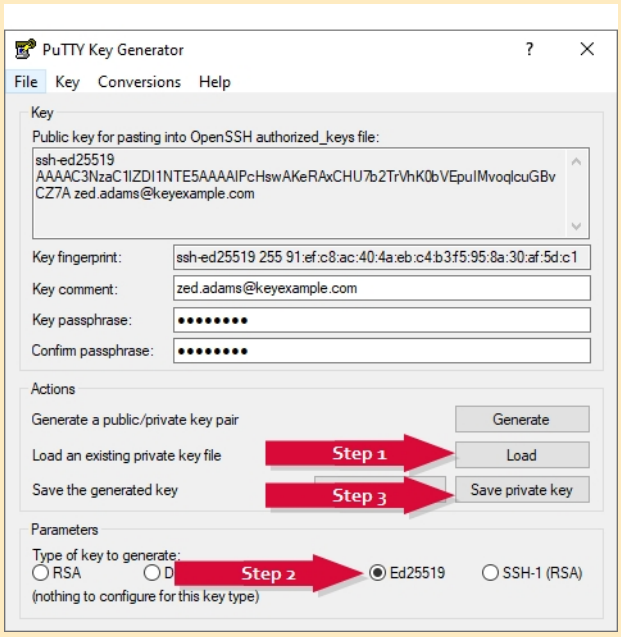


Figure 293: Use PuTTY Key Generator to Convert Zed's Private Key

4. Uses the Keyfactor Command Management Portal to create Linux logons for Zed on each of the three servers that Zed should have access to and map Zed's new public key to these three logons (see Editing Access to an SSH Server Group on page 563).

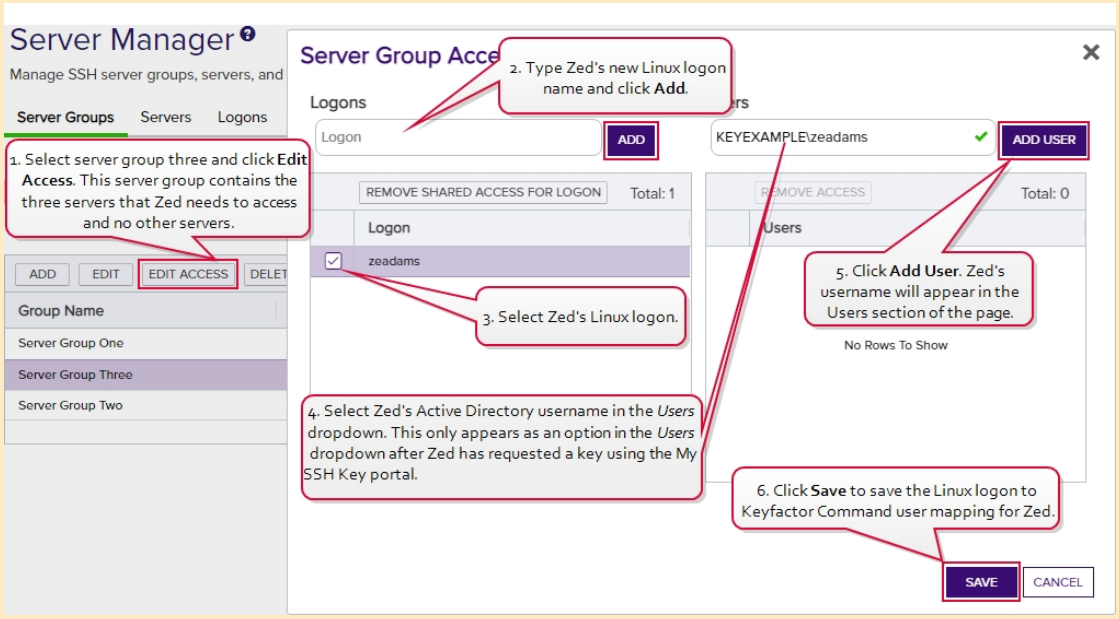


Figure 294: Create Logons and Mappings for Zed

Note: The three servers that Zed needs access to are in a server group so the administrator can create Zed's logons and map his key using the Access Management option on the Server Group page. If the servers were in different server groups or the server group contained servers to which Zed should not have access, the administrator would need to create the logons and mappings separately for each server using the Access Management option on the Servers page (see Editing Access to an SSH Server on page 579).

5. Waits for the logons to be created on the three servers and the public key to be published to them. The time that this takes depends on the frequency of the server group synchronization schedule (see Adding Server Groups on page 561).

6. Instructs Zed to configure PuTTY to use the private key for authentication, providing also connection information for the three Linux servers to which he will be connecting.

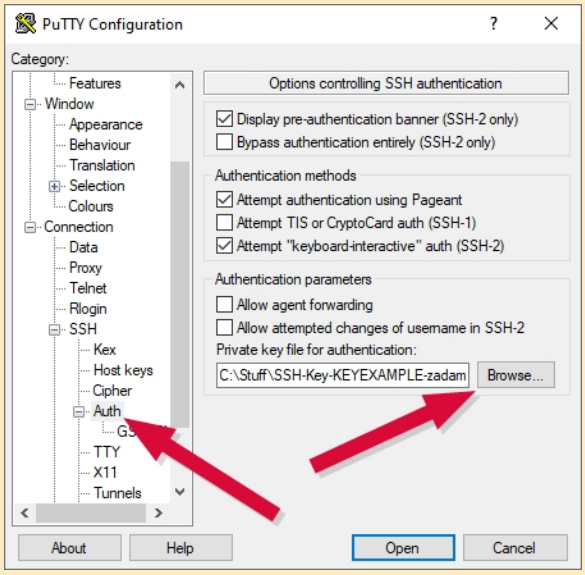


Figure 295: Configure PuTTY to Use Zed's Private Key

7. Confirms that Zed is able to successfully connect using secured SSH to each of the three servers.

This information is included for a key:

Creation Date

The date on which the SSH key pair was generated.

Stale Date

The date on which the SSH key pair is considered to have reached the end of its lifetime. By default, the lifetime of an SSH key pair is 365 days (see Application Settings: SSH Tab on page 620).

Key Type

A number of cryptographic algorithms can be used to generate SSH keys. Keyfactor Command supports RSA, Ed25519, and ECDSA. RSA keys are more universally supported, and this is the default key type when generating a new key.

Key Length

The key length available when generating a new key depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA. The default key length is 2048.

Email

The email address of the user requesting the key. This email address is used to alert the user when the key pair is approaching the end of its lifetime (see Key Rotation Alerts on page 203).

Comment

The user-defined descriptive comment, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks.

SHA256 Fingerprint

The fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.

Public Key

The public key of the key pair.

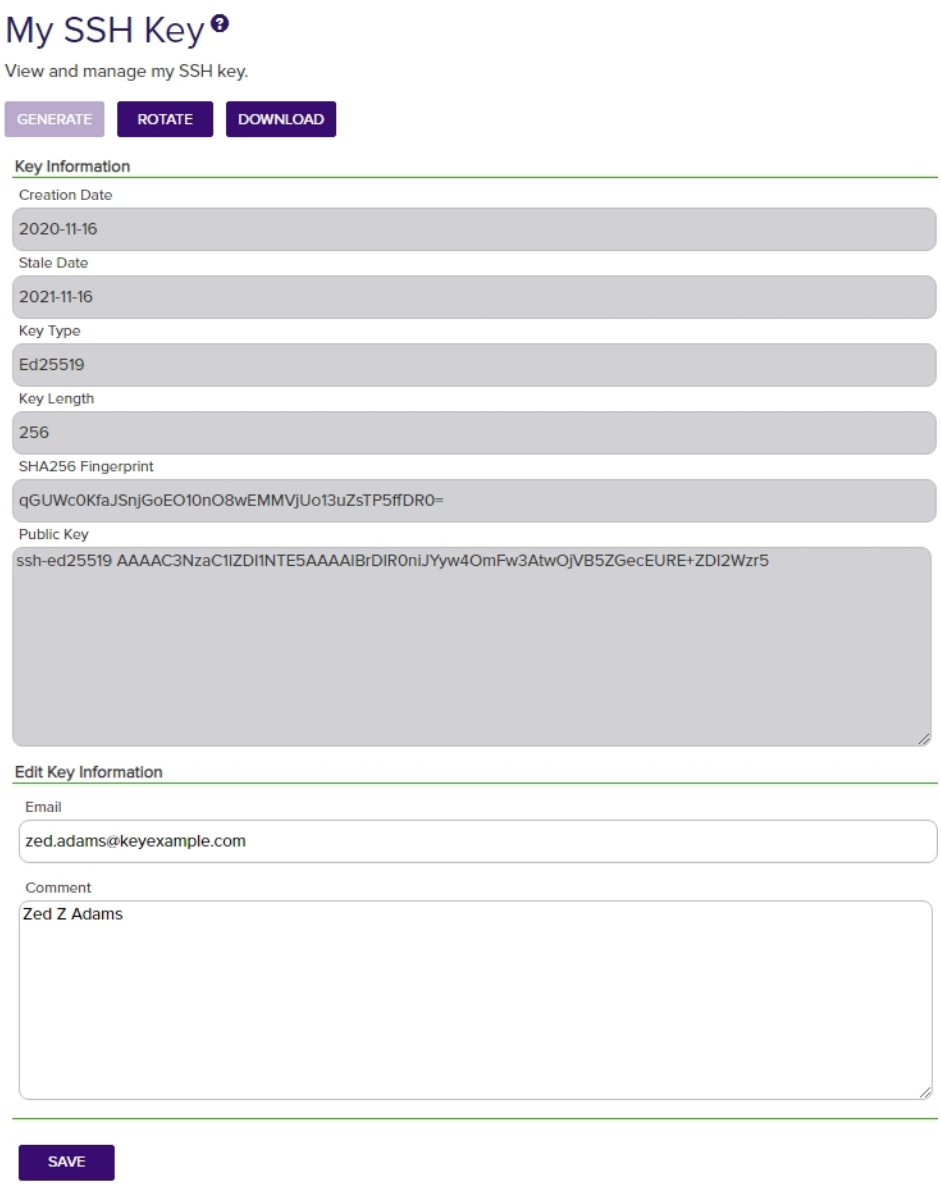


Figure 296: Key Information for an SSH User Key

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Generating a New Key

Important: A given user can only have one SSH key pair in Keyfactor Command. Generating a new key pair removes the existing key pair from Keyfactor Command, if one exists. This means any mappings between the Keyfactor user and Linux logon accounts will be updated with the public key from the new key pair. This essentially invalidates the user's previous private key for servers managed with the Keyfactor Bash Orchestrator. Although the Generate button is not active for users who already have a key pair, the Rotate button will also remove the existing key pair.

To generate a new SSH key pair:

1. In the Management Portal, browse to *SSH > My SSH Key*.
2. On the My SSH Key page, click **Generate**.

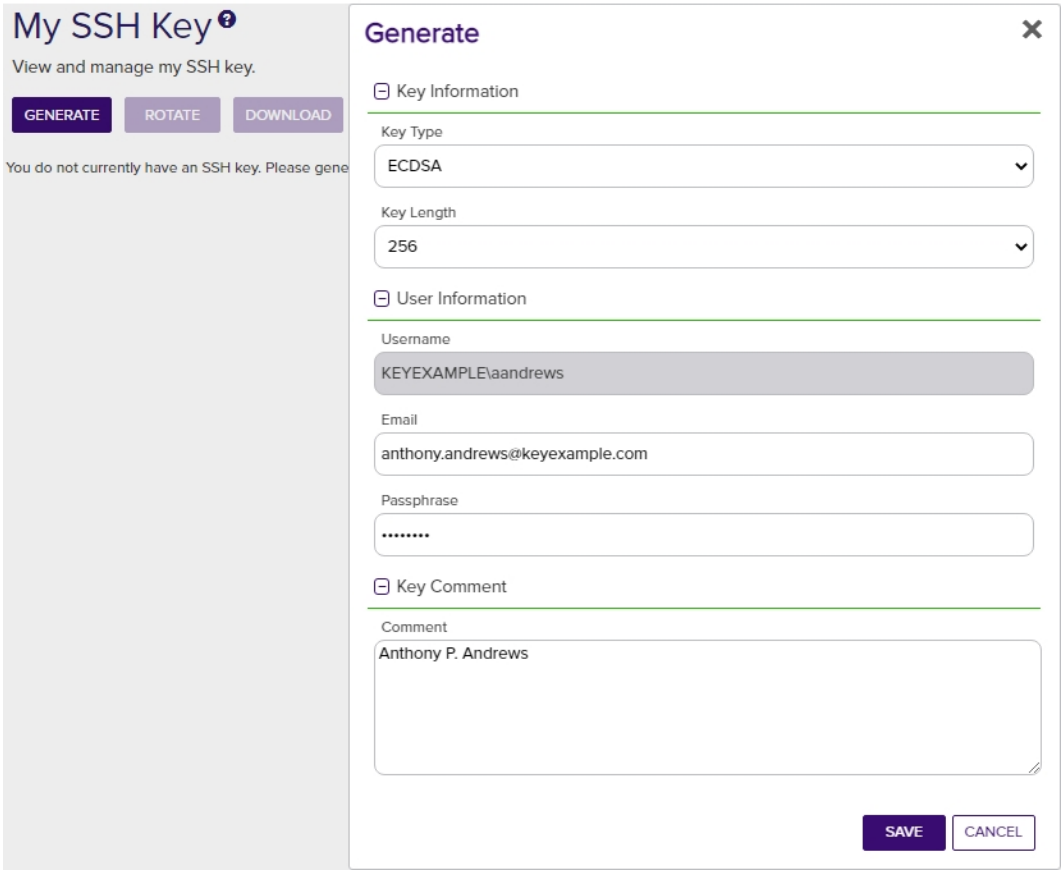


Figure 297: Generate an SSH Key Pair

3. In the Key Information section of the Generate dialog, select a **Key Type** in the dropdown (see Key Type on page 534).

4. In the Key Information section, select a **Key Length** in the dropdown (see Key Length on page 534). The available key lengths will vary depending upon the option selected in the Key Type dropdown.

5. In the User Information section, confirm that the displayed **Username** matches the Active Directory user name you wish to associate with your key. This field defaults to your logged in username and cannot be edited.

6. In the User Information section, enter an **Email** address. This address is used for key rotation alerts (see Key Rotation Alerts on page 203). This field is required.

7. In the User Information section, enter a **Passphrase** to encrypt the downloaded copy of the private key of the key pair. You will need to provide this passphrase again when you use the private key to connect via SSH. By default, the minimum password length is 12 characters (see the *SSH Key Password* setting in Application Settings: SSH Tab on page 620). This field is required.

   Tip: Your private key downloads immediately at the conclusion of the generation process, encrypted with this passphrase. You may later download the private key again from this same page and encrypt it with a different passphrase, if desired.

8. In the Key Comment section, enter a **Comment** to include with the key. This field is optional.

   Tip: Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may contain any characters supported for string fields, including spaces and most punctuation marks.

9. Click **Save** to create the key pair.

Tip: Once the key pair is generated, the user needs to download the private key as an encrypted file and store it locally and an administrator needs to use Keyfactor Command to associate the user's Keyfactor user account with his or her Linux logon account on the target server that the user wishes to access via SSH. After this is complete and the orchestrator has published the user's public key to the target server, the user may connect via SSH to the target server using the new private key for authentication. For more information, see SSH on page 525.

Rotating a Key

The rotate key option is used to replace an existing key that is approaching the end of its life or has been compromised. If key rotation alerts have been configured in the environment (see Key Rotation Alerts on page 203), the user will receive an email when the key is approaching the end if its lifetime to instruct the user to rotate his or her keys.

Important: A given user can only have one SSH key pair in Keyfactor Command. Generating a new key pair with the rotate option removes the existing key pair from Keyfactor Command. This means any mappings between the Keyfactor user and Linux logon accounts will be updated with the public key from the new key pair. This essentially invalidates the user's previous private key for servers managed with the Keyfactor Bash Orchestrator.

The rotate dialog defaults to all the existing settings of the user's current key. At its simplest, users may choose to accept all the defaults, enter a passphrase to encrypt the downloaded private key and click save to generate the new key pair.

To rotate an SSH key pair:

1. In the Management Portal, browse to *SSH > My SSH Key*.
2. On the My SSH Key page, click **Rotate**.

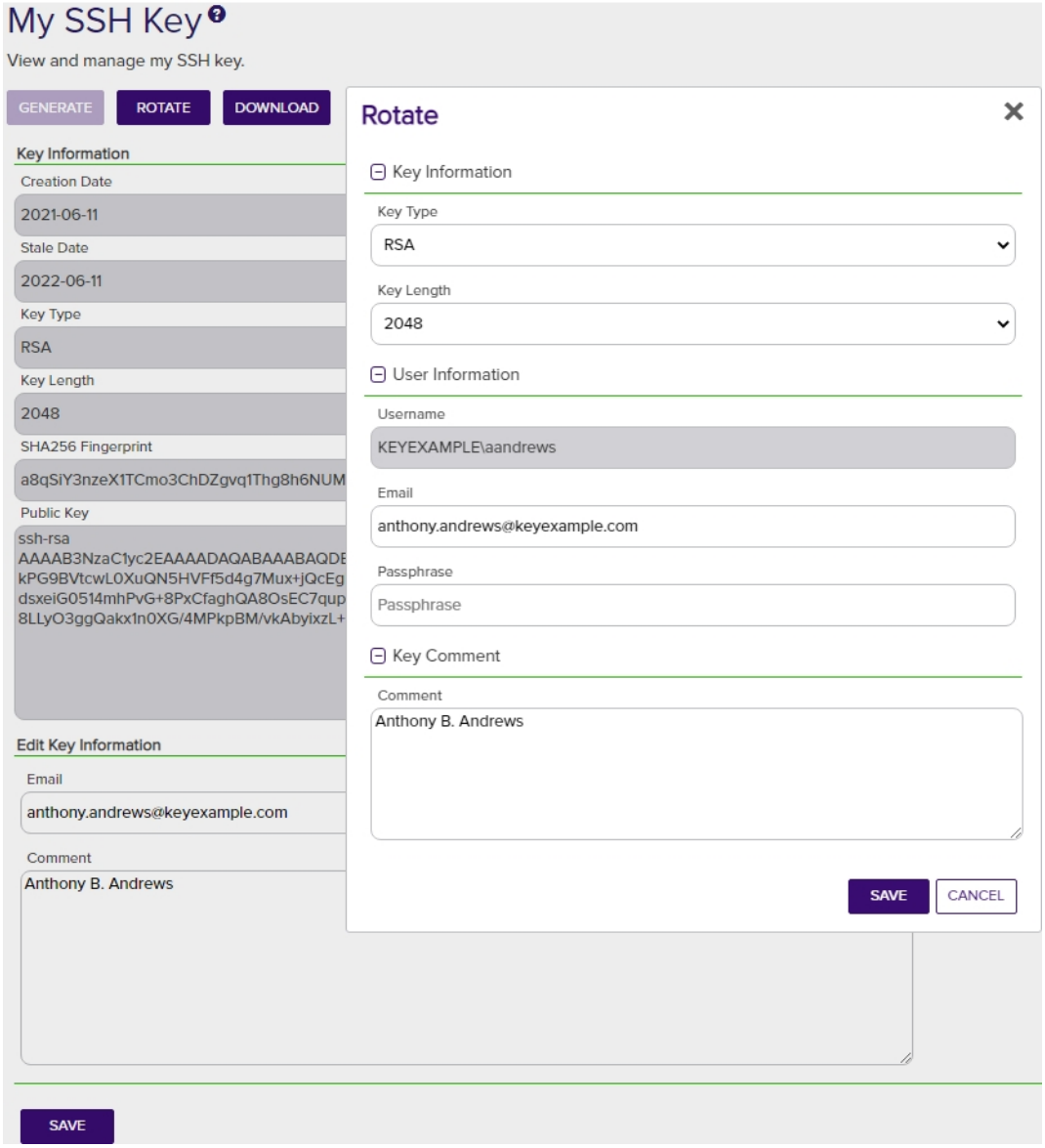


Figure 298: Rotate an SSH Key Pair

3. In the Key Information section of the Rotate dialog, modify the existing **Key Type** in the drop-down, if desired (see Key Type on page 534).

4. In the Key Information section, modify the existing **Key Length** in the dropdown, if desired (see Key Length on page 534). The available key lengths will vary depending upon the option select in the Key Type dropdown.

5. In the User Information section, confirm that the displayed **Username** matches the Active Directory user name you wish to associate with your key. This field defaults to your logged in username and cannot be edited.

6. In the User Information section, modify the existing **Email** address, if desired. This address is used for key rotation alerts (see Key Rotation Alerts on page 203). This field is required.

7. In the User Information section, enter a **Passphrase** to encrypt the downloaded copy of the private key of the key pair. You will need to provide this passphrase again when you use the private key to connect via SSH. By default, the minimum password length is 12 characters (see the *SSH Key Password* setting in Application Settings: SSH Tab on page 620). This field is required.

8. In the Key Comment section, modify the existing **Comment** to include with the key, if desired. This field is optional.

   Tip: Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks.

9. Click **Save** to create the new key pair.

Tip: Once the key pair is generated, the user needs to download the private key as an encrypted file and store it locally and an administrator needs to use Keyfactor Command to associate the user's Keyfactor user account with his or her Linux logon account on the target server that the user wishes to access via SSH. After this is complete and the orchestrator has published the user's public key to the target server, the user may connect via SSH to the target server using the new private key for authentication. For more information, see SSH on page 525.

Downloading a Key

After generating a key pair, you need to download the private key on the machine from which you will be making SSH connections. Although the private key is encrypted, for best security practice it should not be moved around from machine to machine.

The key downloads in the proprietary OpenSSH private key format, encrypted by a user-defined password.

Only the private key can be downloaded with the download option, though the public key is displayed on the screen and may be copied and pasted to a file, if desired.

To download the private key:

1. In the Management Portal, browse to *SSH > My SSH Key*.

2. On the My SSH Key page, confirm that you have been issued a key pair and click **Download**.

3. In the Download dialog, enter a passphrase that will be used to encrypt the private key. By default, the minimum password length is 12 characters (see the *SSH Key Password* setting in Application Settings: SSH Tab on page 620). This field is required.

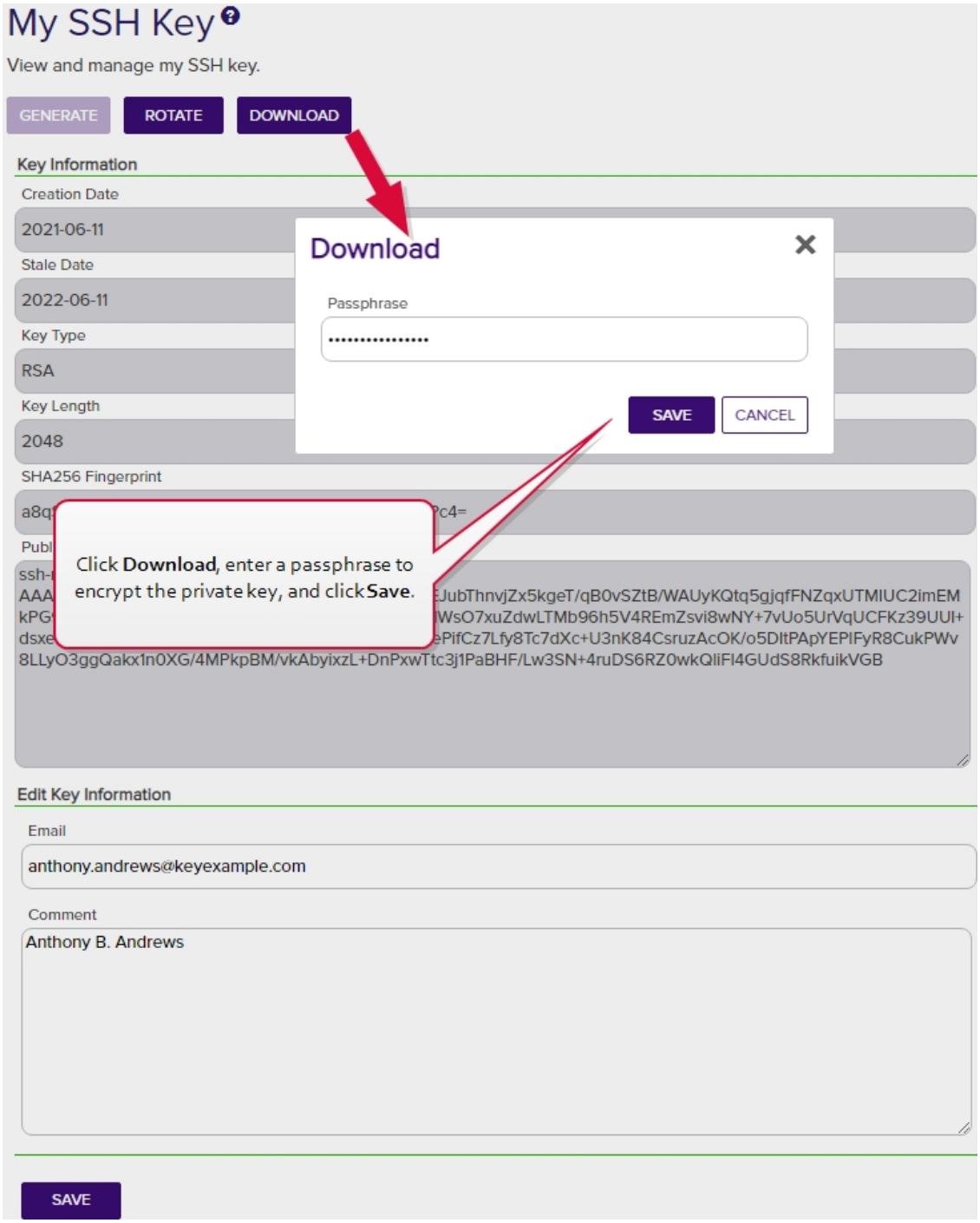


Figure 299: Add a Password to Encrypt the Downloaded Private Key

4. Click **Download** to save the file to your local machine.

By default, the file has the following name, where *DOMAIN* is your Active Directory domain name and *username* is the Active Directory user name of the user logged into the Keyfactor Command Management Portal:

SSH-Key-*DOMAIN*-*username*.identity

Editing Key Information

Once you have generated an SSH key pair, most things about the key pair are fixed and cannot be changed. However, two pieces of key information can be changed for an existing key pair—the email address to which alerts about the key should be directed and the comment associated with the public key.

To modify the email address or key comment:

1. In the Management Portal, browse to *SSH > My SSH Key*.
2. On the My SSH Key page, update the fields in the Edit Key Information section as needed and click **Save**.

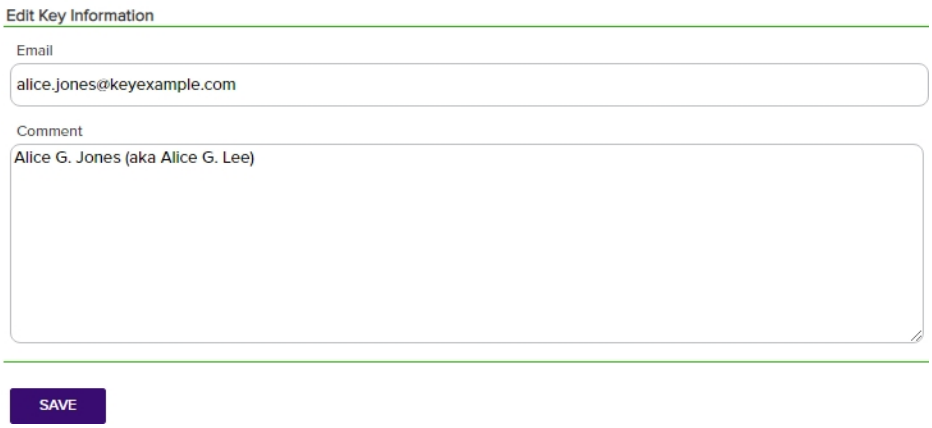


Figure 300: Edit SSH User Key Information

Changes made to the key comment will be published to any associated servers during the next synchronization cycle.

2.1.10.2 Service Account Keys

On the Service Account Keys page, an administrator can view and download existing keys issued for service accounts and generate new key pairs.

Example: An administrator wants to generate a new SSH key pair for the green chicken application, which is a Linux-based log aggregation application. The application uses secure

SSH to communicate internally between the server collecting the logs and the servers from which the logs are being collected. All the servers are controlled by the Keyfactor Bash Orchestrator. The servers are set to both inventory and publish policy. To accomplish this, the administrator:

1. Uses the Keyfactor Command Management Portal to create a new key pair (see Creating a Service Account Key on page 545). She enters the following information in the form:
    - **Key Type**: Ed25519
    - **Key Length**: 256
    - **Server Group**: Server Group One

      The server group to which the Linux servers belong that the public key will be distributed to.
    - **Client Hostname**: appsrvr75

      The Linux server on which the private key of the SSH key pair will be download. This does not need to be a server added for management in Keyfactor Command and is a field for reference only.
    - **Username**: svc_greenchicken

      The service account name the application uses. This does not need to match the Linux logon name the application uses. This username together with the client hostname make the full user name for the service account key within Keyfactor Command svc_greenchicken@appsrvr75.
    - **Email**: pkiadmins@keyexample.com

      The group responsible for rotating the key when it reaches the end of its lifetime. This group will receive email alerts when the key is becoming stale.
    - **Passphrase**: A complex password used to secure the private key.

      She needs to record the passphrase because this will be needed by application to access the private key.
    - **Comment**: Green Chicken Service

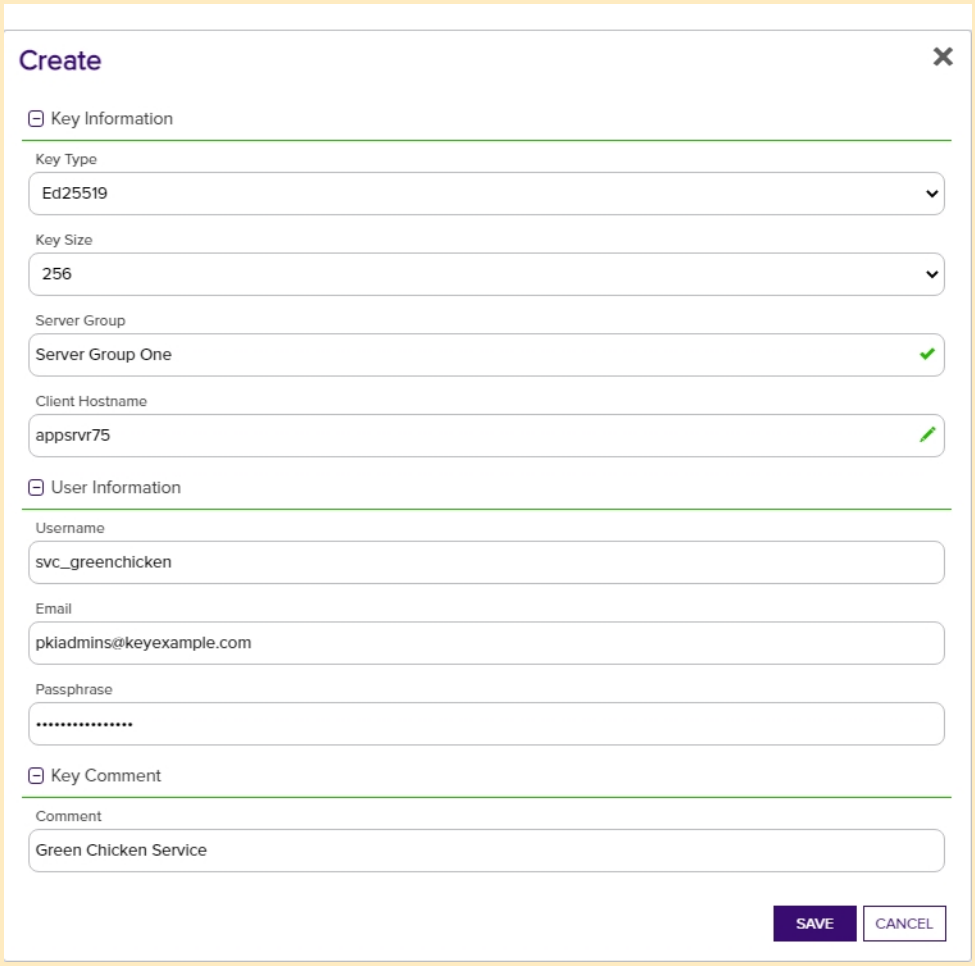


Figure 301: Acquire a New Service Account Key

2. Downloads the SSH private key on the server doing the log collection, from which the SSH connections will be made to collect logs.

3. Uses the Management Portal to map the new public key for the full service account user name (svc_greenchicken@appsrvr75) to the Linux logons for the service on the servers from which the logs will be collected (see Editing Access to an SSH Server Group on page 563).

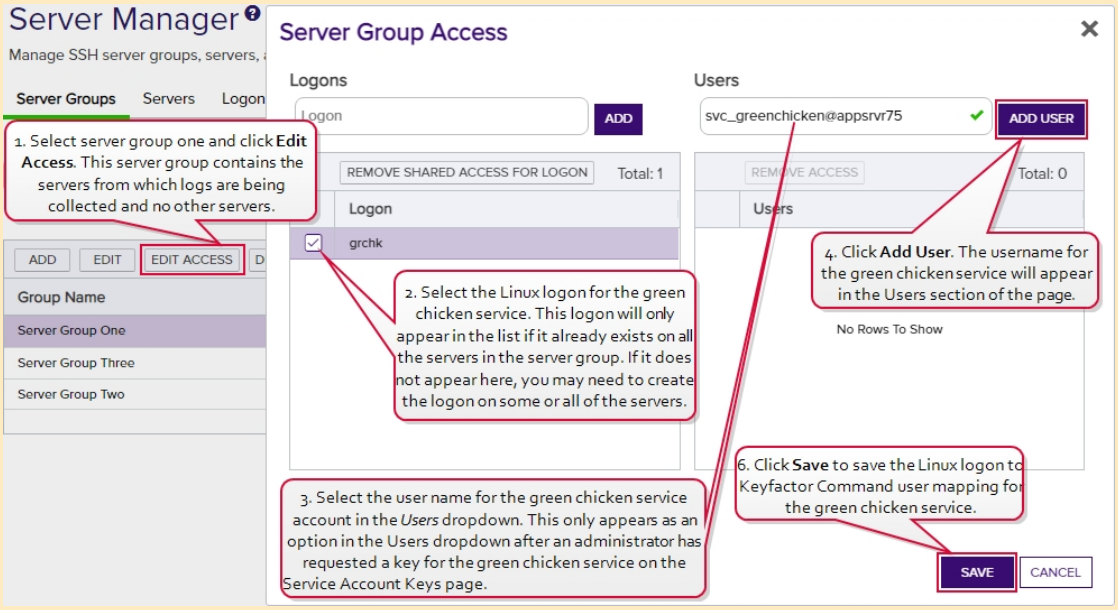


Figure 302: Map Service Account Public Key to Logon

Note: The servers that the logs will be collected from are organized into a server group so the administrator can create logons and map the service account key using the Access Management option on the Server Group page. If the servers were in different server groups or the server group contained servers which should not be updated with logons and keys for the green chicken service, the administrator would need to create the logons and mappings separately for each server using the Access Management option on the Servers page (see Editing Access to an SSH Server on page 579).

4. Waits for the public key to be published to the servers. The time that this takes depends on the frequency of the server group synchronization schedule (see Adding Server Groups on page 561).
5. Confirms that the service is able to successfully connect using secured SSH.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Creating a Service Account Key

To create a new service account key:

1. In the Management Portal, browse to *SSH > Service Account Keys*.
2. On the Service Account Keys page, click **Create**.

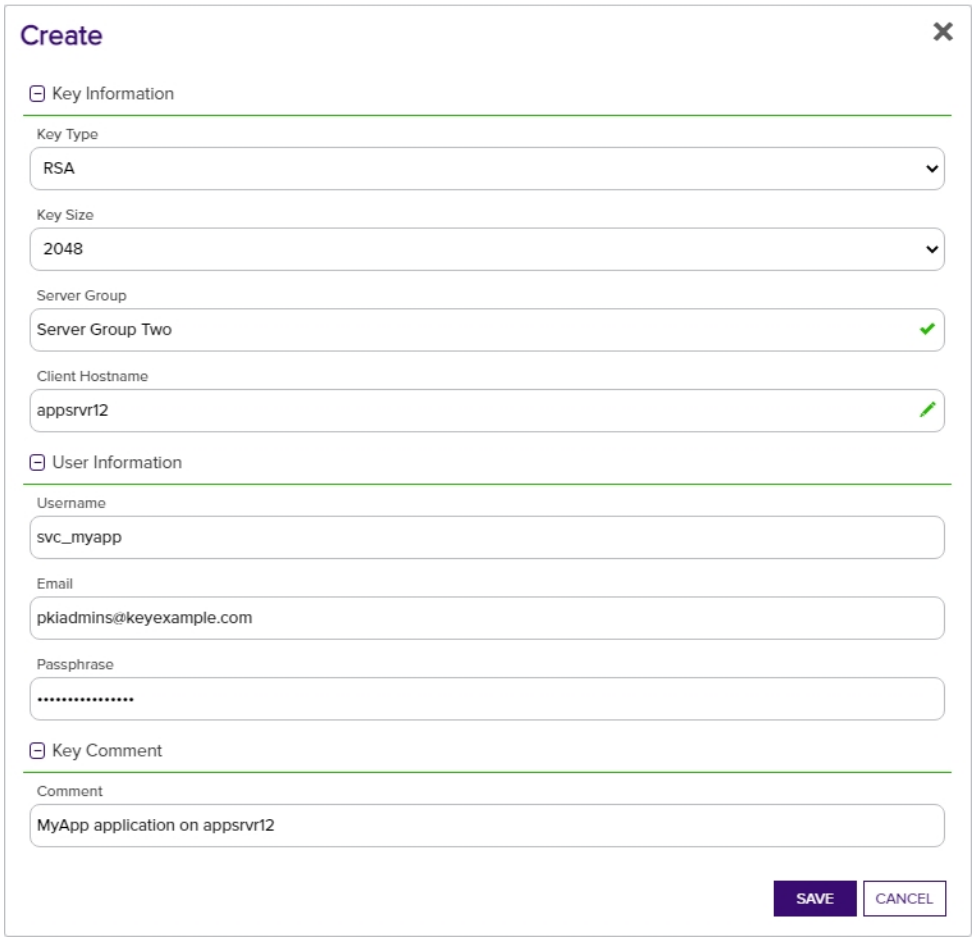


Figure 303: Add a Service Account Key

3. In the Key Information section of the Create dialog, select a **Key Type** in the dropdown (see Key Type on page 534).
4. In the Key Information section, select a **Key Length** in the dropdown (see Key Length on page 534). The available key lengths will vary depending upon the option select in the Key Type dropdown.
5. In the Key Information section, select a **Server Group** in the dropdown (see SSH Server Groups on page 560). The server group is used to control who has access in the Management Portal to the service account key. It does not limit where the key can be published. This field is required.
6. In the Key Information section, enter a **Client Hostname** reference for the service account key. This field is used for reference only and does not need to match an actual client hostname. It is

used when building the full user name of the service account key for mapping to Linux logons for publishing to Linux servers (e.g. username@client_hostname). The naming convention is to enter the hostname of the server on which the application that will use the private key resides (e.g. appsrvr12), but you can put anything you like in this field (e.g. cheesetoast). This field is required.

7. In the User Information section of the page, enter the **Username** of the service account that will be using the key to connect to the target server (e.g. svc_myapp). This username will be combined with the Client Hostname to build the full user name of the service account key for mapping to Linux logons (e.g. svc_myapp@appsrvr12). You will need to know this full user name when creating the mappings to publish the public key to the target servers (see Editing Access to an SSH Server Group on page 563, Editing Access to an SSH Server on page 579, Adding Logons on page 585, or Editing or Deleting a Logon on page 588). This field is required.

8. In the User Information section of the page, enter the **Email** address of the administrator or group of administrators responsible for managing the key. This is the address to which key rotation alerts for this key will be directed (see Key Rotation Alerts on page 203). This field is required.

9. In the User Information section, enter a **Passphrase** to encrypt the downloaded copy of the private key of the key pair. The service that uses the private key will need to be able to provide it when connecting via SSH. By default, the minimum password length is 12 characters (see the *SSH Key Password* setting in Application Settings: SSH Tab on page 620). This field is required.

   Tip: The private key downloads immediately at the conclusion of the creation process, encrypted with this passphrase. You may later download the private key again from this same page and encrypt it with a different passphrase, if desired.

10. In the Key Comment section, enter a **Comment** to include with the key. This field is optional.

    Tip: Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may contain any characters supported for string fields, including spaces and most punctuation marks.

11. Click **Save** to save the new service account key.

    Tip: Once the key pair is generated, an administrator needs to download the private key as an encrypted file and store it locally on the machine from which the service will make SSH connections using the private key. Additionally, an administrator needs to use Keyfactor Command to map the full user name built from the username and client hostname entered when generating the service account key pair (e.g. svc_myapp@appsrvr12) to the Linux logon account that the service account will operate as when logging in via SSH on the target server(s) where the public key needs to reside (see Editing Access to an SSH Server Group on page 563, Editing Access to an SSH Server on page 579, Adding Logons on page 585, or Editing or Deleting a Logon on page 588). After this is complete and the orchestrator has published the public key to the target

server(s), the service may connect via SSH to the target server(s) using the new private key for authentication. For more information, see SSH on page 525.

Editing Service Account Key Information

Once you have generated an SSH key pair, most things about the key pair are fixed and cannot be changed. However, two pieces of key information can be changed for an existing key pair—the email address to which alerts about the key should be directed and the comment associated with the public key.

Tip: Only service account keys belonging to server groups that the current user is the owner on appear in the grid unless the user holds the *SSH Enterprise Admin* role.

To modify the email address or key comment:

1. In the Management Portal, browse to *SSH* > *Service Account Keys*.
2. On the Service Account Keys page, double-click the key for the desired service account in the grid, highlight the row in the grid and click **Edit** at the top of the grid, or right-click the key in the grid and choose **Edit** from the right-click menu.
3. In the Edit Key dialog, update the **Email** and **Comment** fields as needed and click **Save**.

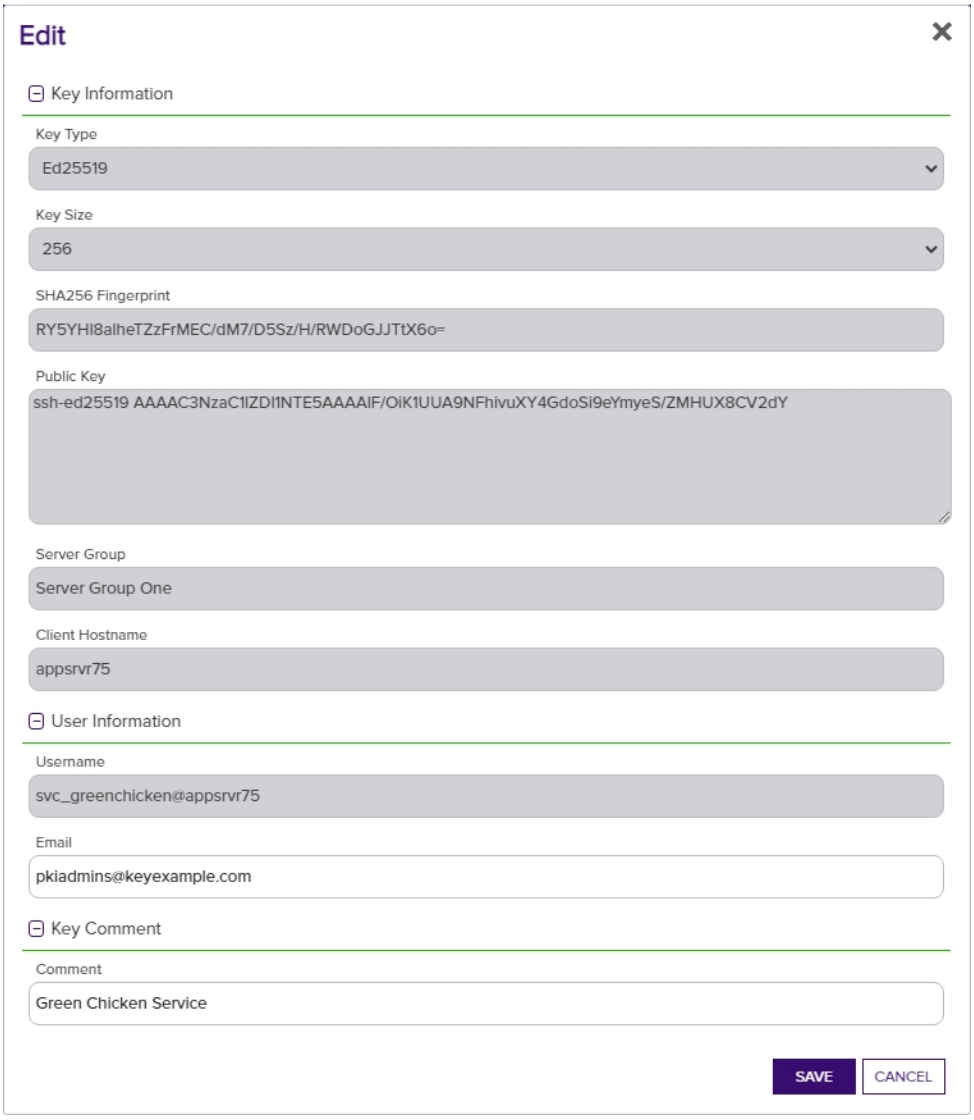


Figure 304: Edit SSH Service Account Key Information

Changes made to the key comment will be published to any mapped logons on associated servers during the next synchronization cycle.

Rotating a Service Account Key

The rotate key option is used to replace an existing key that is approaching the end of its life or has been compromised. If key rotation alerts have been configured in the environment (see Key Rotation Alerts on page 203), the administrator responsible for managing the service account key will receive an email when the key is approaching the end if its lifetime to instruct the him or her to rotate the service account key.

Important: A given service account can only have one SSH key pair in Keyfactor Command. Generating a new key pair with the rotate option removes the existing key pair from Keyfactor Command. This means any mappings between the Keyfactor service account and Linux logon accounts will be updated with the public key from the new key pair. This essentially invalidates the service account's previous private key for servers managed with the Keyfactor Bash Orchestrator.

The rotate dialog defaults to all the existing settings of the service account's current key. At its simplest, the administrator may choose to accept all the defaults, enter a passphrase to encrypt the downloaded private key and click save to generate the new key pair.

To rotate a service account key pair:

1. In the Management Portal, browse to *SSH > Service Account Keys*.
2. On the Service Account Keys page, click **Rotate**.

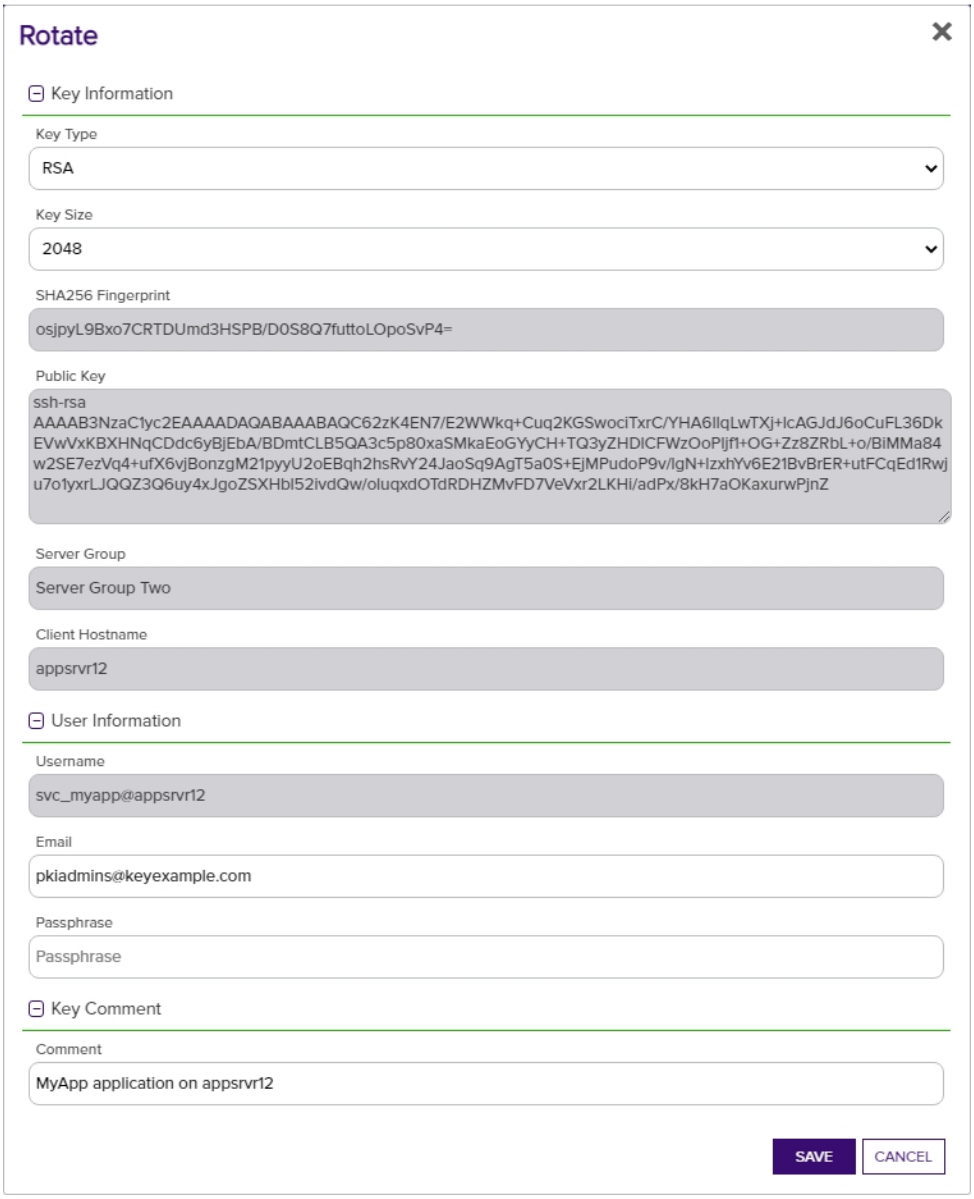


Figure 305: Rotate an SSH Key Pair

3. In the Key Information section of the Rotate dialog, modify the existing **Key Type** in the dropdown, if desired (see Key Type on page 534).

4. In the Key Information section, modify the existing **Key Length** in the dropdown, if desired (see Key Length on page 534). The available key lengths will vary depending upon the option select in the Key Type dropdown.

5. In the User Information section, modify the existing **Email** address, if desired. This address is used for key rotation alerts (see Key Rotation Alerts on page 203). This field is required.

6. In the User Information section, enter a **Passphrase** to encrypt the downloaded copy of the private key of the key pair. You will need to provide this passphrase again when you use the private key to connect via SSH. By default, the minimum password length is 12 characters (see the *SSH Key Password* setting in Application Settings: SSH Tab on page 620). This field is required.

7. In the Key Comment section, modify the existing **Comment** to include with the key, if desired. This field is optional.

   Tip: Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks.

8. Click **Save** to create the new key pair.

Tip: Once the key pair is generated, an administrator needs to download the private key as an encrypted file and store it locally on the machine from which the service will make SSH connections using the private key. Additionally, an administrator needs to use Keyfactor Command to map the full user name built from the username and client hostname entered when generating the service account key pair (e.g. svc_myapp@appsrvr12) to the Linux logon account that the service account will operate as when logging in via SSH on the target server(s) where the public key needs to reside (see Editing Access to an SSH Server Group on page 563, Editing Access to an SSH Server on page 579, Adding Logons on page 585, or Editing or Deleting a Logon on page 588). After this is complete and the orchestrator has published the public key to the target server(s), the service may connect via SSH to the target server(s) using the new private key for authentication. For more information, see SSH on page 525.

Deleting a Service Account Key

To delete a service account key, highlight the row in the service account keys grid and click **Delete** at the top of the grid or right-click the key in the grid and choose **Delete** from the right-click menu.

Tip: Only service account keys belonging to server groups that the current user is the owner on appear in the grid unless the user holds the *SSH Enterprise Admin* role.

Downloading a Service Account Key

After generating a key pair, you need to download the private key on the machine from which you will be making SSH connections. Although the private key is encrypted, for best security practice it should not be moved around from machine to machine.

The key downloads in the proprietary OpenSSH private key format, encrypted by a user-defined password.

Only the private key can be downloaded with the download option, though the public key is displayed in the edit dialog and may be copied and pasted to a file, if desired.

Tip: Only service account keys belonging to server groups that the current user is the owner on appear in the grid unless the user holds the *SSH Enterprise Admin* role.

To download the private key:

1. In the Management Portal, browse to *SSH > Service Account Keys*.
2. On the Service Account Keys page, locate the key for the desired service account and click **Download**.

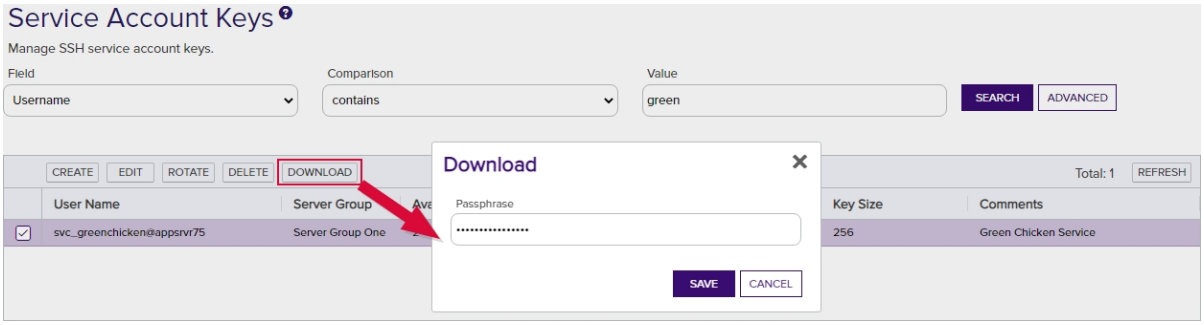


Figure 306: Download a Service Account Private Key

3. In the Download dialog, enter a passphrase that will be used to encrypt the private key. By default, the minimum password length is 12 characters (see the *SSH Key Password* setting in Application Settings: SSH Tab on page 620). This field is required.
4. Click **Download** to save the file to the local machine.

   By default, the file has the following name:

   SSH-Key-Service-Account.identity

Using the Service Account Key Search

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Tip: Only service account keys belonging to server groups that the current user is the owner on appear in the grid unless the user holds the *SSH Enterprise Admin* role.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

ServerGroupName

Complete or partial matches with the name of the server group that the service account key is associated with.

Username

Complete or partial matches with the username of the service account key. The username is made up of the username and client hostname entered when the service account key was created (e.g. myapp@appsrvr75).

CreationDate

The date on which the key was created.

KeyType

Whether the key is RSA, ECC, or Ed25519

KeyLength

The key size of the key.

Comments

Complete or partial matches with the user-defined comments on the key.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

2.1.10.3 Unmanaged SSH Keys

When your SSH servers are configured in inventory only mode doing discovery, keys discovered on the servers are considered unmanaged and are displayed on the Unmanaged Keys page.

On this page you can review the discovered keys to get a sense of what's out there. You can view the keys, key comments, fingerprint, type and length. Once you switch your servers to inventory and publish policy mode, deleting a key from the unmanaged keys page will also delete the key from the server(s) in this mode on which it is found.

Note: Deleting a key on this page when the associated server is still in inventory only mode will *not* delete the key on the target server. The next time the server is scanned, the key will re-appear in Keyfactor Command.

As you bring your servers under management, clean up old keys, and control installation of new keys, the number of keys appearing on the unmanaged keys page should begin to diminish. Eventually, the page should be empty when all your servers have been brought under management and all old keys have been replaced with new, managed, keys.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Viewing Unmanaged SSH Keys

To view details for an unmanaged public key, double-click the key, right-click the key and choose **View** from the right-click menu, or highlight the row in the unmanaged keys grid and click **View** at the top of the grid.

The view dialog includes two tabs:

- On the Basic tab, you can see information about the key itself, including the key length, fingerprint, comments associated with the key, and the public key itself.

Figure 307: View Basic Tab of an Unmanaged SSH Key

- On the Logon tab, you can view Linux logon names and servers mapped to the public key.

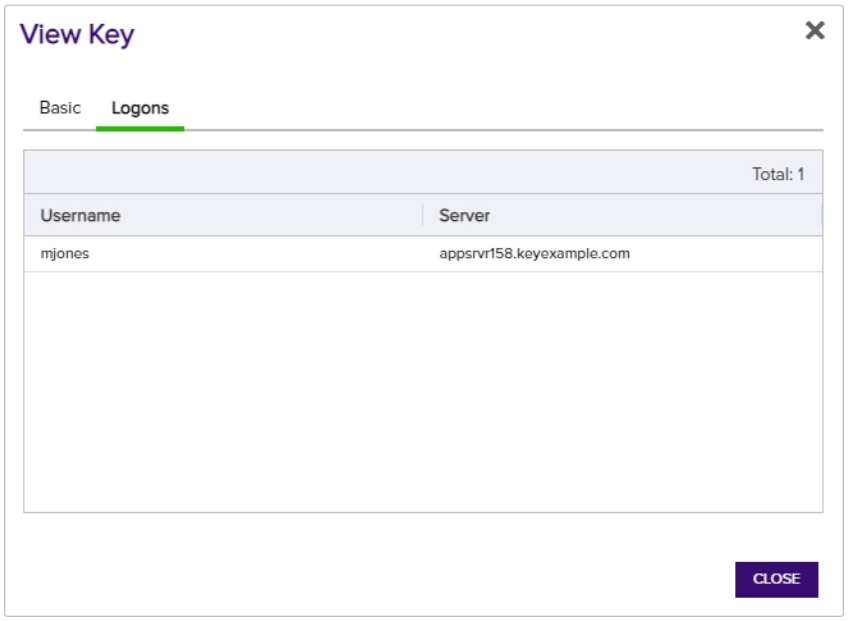


Figure 308: View Logon Tab of an Unmanaged SSH Key

Deleting an Unmanaged Key

To delete an unmanaged key, highlight the row in the unmanaged keys grid and click **Delete** at the top of the grid or right-click the key in the grid and choose **Delete** from the right-click menu.

Note: When you delete an unmanaged key that's found on any servers operating in inventory and publish policy mode (see SSH on page 525), the key will be removed from the target servers as well as from Keyfactor Command.

Using the Unmanaged Keys Search

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

DiscoveredDate

The date on which the key was discovered.

KeyLength

The key size of the key.

KeyComments

Complete or partial matches with the user-defined comments on the key.

KeyType

Whether the key is RSA, ECC, or Ed25519

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is null (-eq NULL)

- Is not equal to (-ne)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

In addition to the options available in the query builder, three special values can be used in selected searches by typing them in directly:

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).

- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

Important: The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

2.1.10.4 Server Manager

SSH key management is performed by one or more Keyfactor Bash Orchestrators controlling multiple targets. These are referred to collectively in the Management Portal as SSH servers. The SSH servers are collected together into one or more server groups. On the Server Manager page you first create one or more server groups to organize and set policies for your Linux SSH servers and then add an SSH server entry for each server you want to control with the orchestrator.

Scanning jobs are configured at the server group level. You can toggle between *inventory only* mode and *inventory and publish policy* mode at either the server group level or on an individual server basis, though if a server group is in *inventory and publish policy* mode (configured to *Enforce Publish Policy*), servers in this group cannot be in *inventory only* mode.

Scanning of targets cannot take place until they have been set up for control by the orchestrator (see Install Remote Control Targets on page 3000 in the *Keyfactor Orchestrators Installation and Configuration Guide*).

Tip: If you plan to scan and manage your orchestrator machine(s) in addition to any targets, you will need to add SSH server entries for these as though they were targets.

Once the scanning has begun, you can look at the Logons tab to see discovered logons from the targets and associated SSH public keys.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

SSH Server Groups

On the Server Groups tab of the Server Manger page you create server groups that allow you to organize SSH servers and set synchronization schedules and management policies on a group level.

You must create at least one server group before you can add SSH servers into the Keyfactor Command Management Portal.

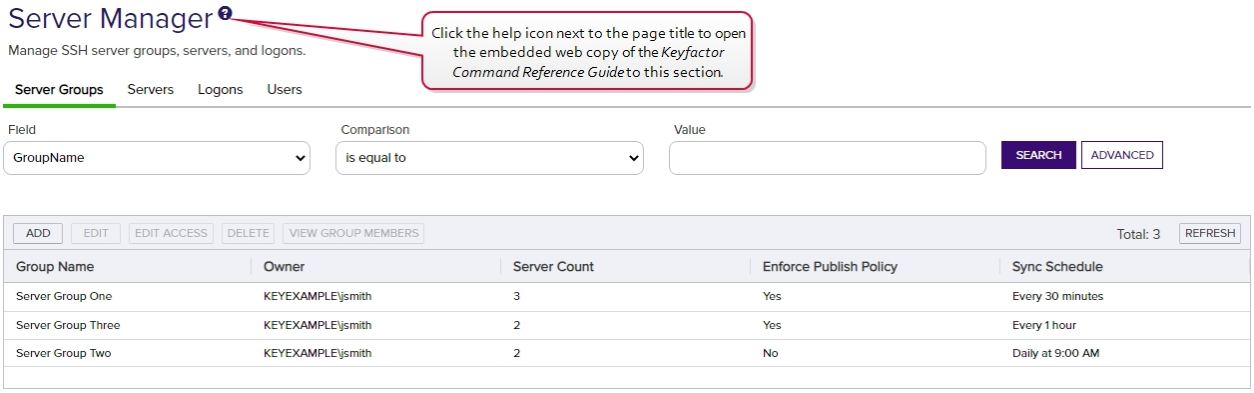


Figure 309: SSH Server Groups Grid

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Reference Guide* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Adding Server Groups

To add a new server group:

1. In the Management Portal, browse to *SSH > Server Manager*.
2. On the Server Manager page, select the Server Groups tab (the default when you first visit the page).
3. On the Server Groups tab, click **Add**.

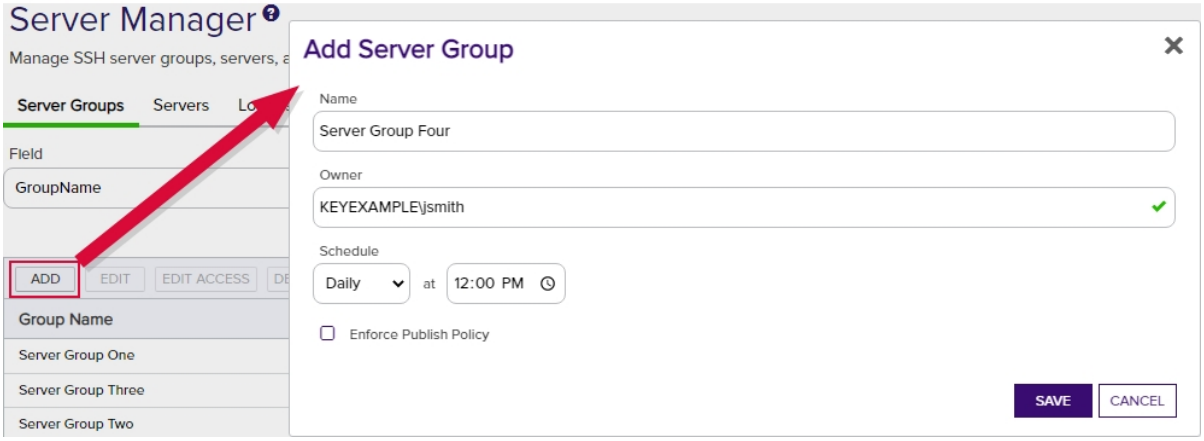


Figure 310: Add a Server Group

4. In the Add Server Group dialog, enter a name for the group in the **Name** field.

5. In the **Owner** dropdown, enter or select an Active Directory user with access to the Keyfactor Command Management Portal holding either the SSH Server Admin or SSH Enterprise Admin role (see SSH Permissions on page 597). Any users with one of these roles who have previously been made an owner on a server group or enrolled for an SSH key (see My SSH Key on page 531) will appear in the Owner dropdown.

6. In the **Schedule** dropdown, select a frequency for the server synchronization job. Possible options are:

    - Interval—Enter an interval from every 1 minute to every 12 hours
    - Daily—Enter selected time
    - Weekly—Enter a selected day or days of the week at a selected time
    - Monthly—Enter a selected day of the month (1st through 27th) at a selected time

    Tip: During initial configuration, you may want to set a short timeframe for job frequency and then extend it as the servers settle into a management routine.

7. If desired, check the **Enforce Publish Policy** box to set the server group to *inventory and publish policy* mode (see SSH on page 525).

8. Click **Save** to save the new server group.

Editing or Deleting a Server Group

To edit a server group, double-click the group, right-click the group and choose **Edit** from the right-click menu, or highlight the row in the server groups grid and click **Edit** at the top of the grid.

Tip: The owner can only be changed by a Keyfactor Command user who holds the *SSH Enterprise Admin* role (see SSH Permissions on page 597).

To delete a server group, highlight the row in the server groups grid and click **Delete** at the top of the grid or right-click the group in the grid and choose **Delete** from the right-click menu.

Editing Access to an SSH Server Group

Using the Edit Access function you create mappings between Keyfactor Command user accounts associated with SSH keys and Linux logons in order to publish the SSH public keys to all the Linux servers that belong to the selected server group (see SSH on page 525). You can also remove the mappings from here, which causes the SSH public keys to be removed from the Linux servers belonging to the selected server group.

Before adding a logon to user mapping, be sure that you have switched either the server group or all servers in the group to which you will add your mapping to *inventory and publish policy* mode (see Server Manager on page 560) so that the key for the user will be published to the servers in the group. If the servers in the server group are in *inventory only* mode and you add a mapping for it in Keyfactor Command, the mapping will appear in Keyfactor Command only and the key for the user will not be published out to the servers in the server group. If only some servers in the server group are in *inventory and publish policy* mode, the key for the user will only be published to those servers.

Tip: The time it will take for changes to access mappings to appear on your Linux servers will depend on the frequency of the server synchronization configured for the server group (see Adding Server Groups on page 561).

To edit the access for a server group, create a mapping between a Linux logon and a Keyfactor Command user, and publish the user's key to all the SSH servers belonging to that server group:

1. In the Management Portal, browse to *SSH > Server Manager*.
2. On the Server Manager page, select the Server groups tab.
3. In the Server groups grid, locate the server group that contains the servers you wish to publish an SSH key to by mapping a Keyfactor Command user to a Linux logon on that server group.
4. Right-click the server group and choose **Edit Access** from the right-click menu or highlight the row in the server groups grid and click **Edit Access** at the top of the grid.

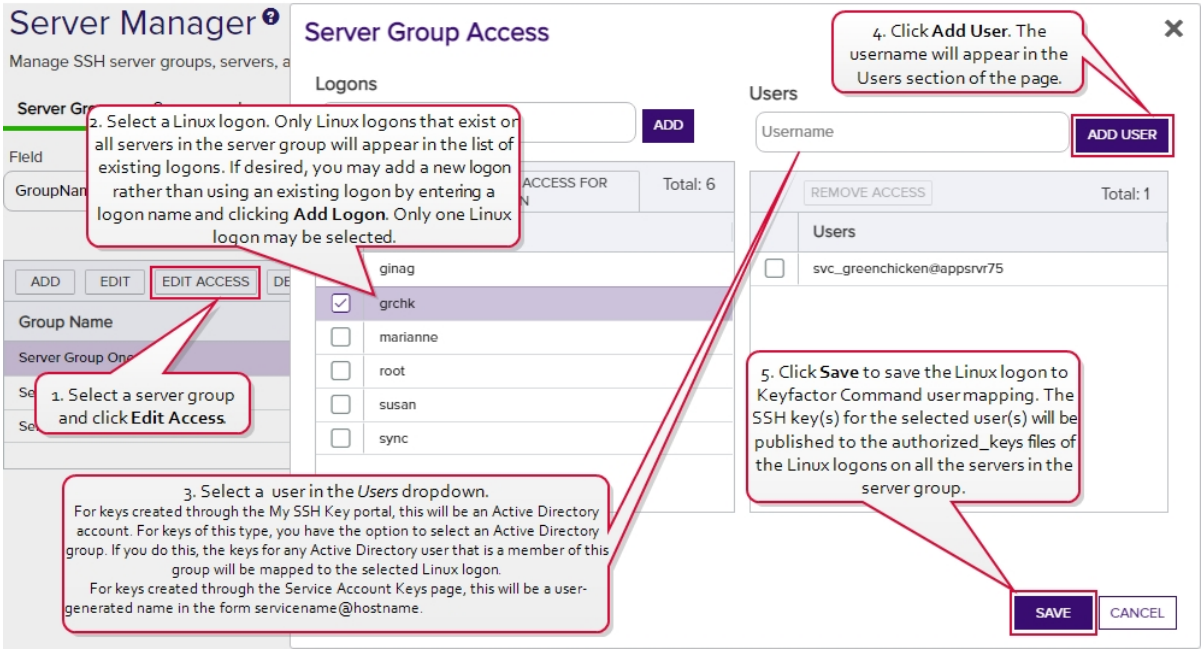


Figure 311: Edit Access for an SSH Server Group

5. On the Access Management page, select an existing Logon on the left side of the page. Logons only appear here if they exist with the same spelling on all servers in the server group. If you wish to add a new logon, enter the new logon name in the Logon field at the top of the left side of the page and click **Add Logon**. The new logon appears at the bottom of the Logon list. Click the **Logon** list title to sort the list, if desired. Select the new logon. Only one logon may be selected.

   Tip: If you have enabled SSSD support for your Keyfactor Bash Orchestrator and are adding a domain user, specify the user in username@domain format. For example bbrown@keyexample.com (or, depending on SSSD configuration, such as the case-sensitivity setting; BBROWN@keyexample.com). Note that the logon may be modified by the SSSD configuration file in ways in which Keyfactor Command cannot know about. Refer to SSH-SSSD Case Sensitivity Flag on page 791 for guidance on what to enter based on how the SSSD case sensitivity flag is configured.

6. In the Users dropdown at the top of the right side of the page, select a *user* or *service account* to associate the logon with. Only Keyfactor users that have keys stored in Keyfactor Command, that have been designated as server group owners, or AD users or groups that have been previously entered for association with a logon will appear in the dropdown. If desired, you may enter an Active Directory user or group name in this field. Using an Active Directory group to create Linux logon to Keyfactor user mappings will cause the keys stored in Keyfactor Command for any Active Directory users that are members of this group to be mapped to the selected Linux logon and published to the servers on which the Linux logon exists. Any Active Directory users that are members of this group but who do not have keys stored in Keyfactor Command will not be

mapped to the selected Linux logon. Click **Add User**.

> **Tip:** For keys created through the My SSH Key portal (see My SSH Key on page 531), a Keyfactor user is an Active Directory user account. For keys created through the Service Account Keys page (see Service Account Keys on page 542), a Keyfactor user is a user-generated service account name of the form servicename@hostname.

7. Repeat step 6 for any other user or service accounts that you wish to map to this logon on the servers in this server group.
8. Click **Save**.

To remove a mapping of a Linux logon to a Keyfactor Command user for all the servers in a server group, remove the public key from the Linux logon's authorized_keys files:

1. In the Management Portal, browse to *SSH > Server Manager*.
2. On the Server Manager page, select the Server Groups tab.
3. In the Server Groups grid, locate the server group that contains the servers you wish to remove an SSH key from by unmapping a Keyfactor Command user from a Linux logon on that server group.
4. Right-click the server group and choose **Edit Access** from the right-click menu or highlight the row in the server groups grid and click **Edit Access** at the top of the grid.

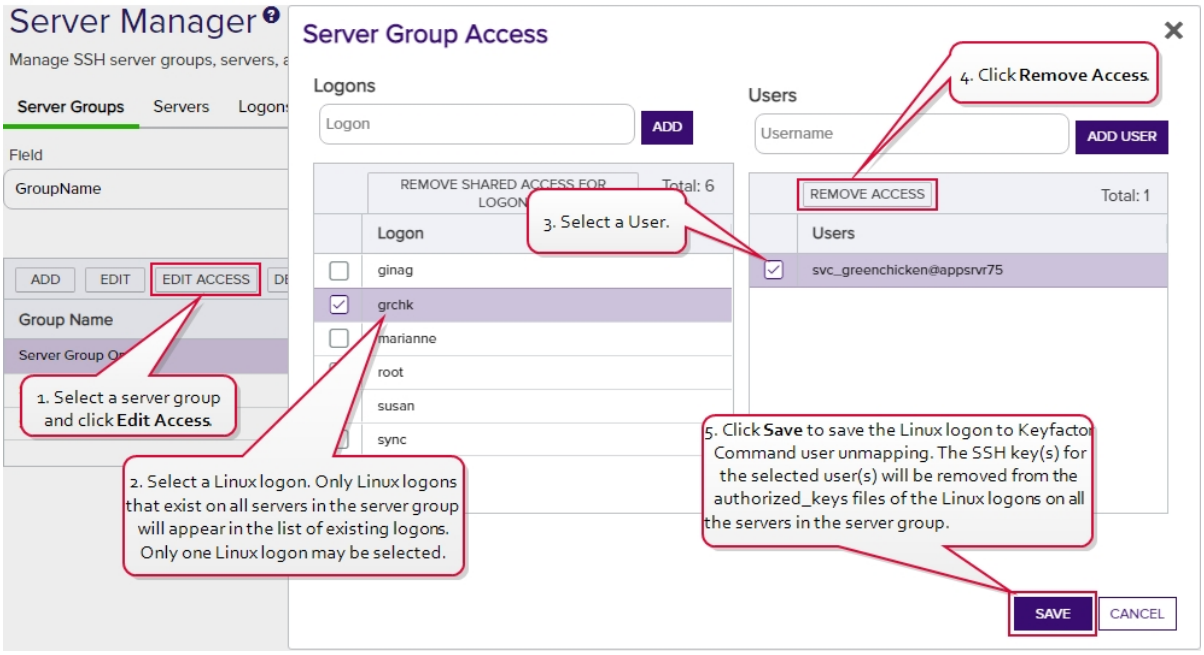


Figure 312: Edit Access for an SSH Server

5. On the Access Management page, select a Logon on the left side of the page. Only one logon may be selected.

6. In the Users section on the right side of the page, select a *user* or *service account* to unmap from the logon. Click **Remove Access** under *Users*. The Linux logon to Keyfactor user mapping for the *selected user* will be removed and the user's SSH key will be removed from the authorized_keys files of the Linux logons on all the servers in the server group.

   Tip: Clicking **Remove Shared Access for Logon** on the *Logons* side of the page removes *all* Linux logon to Keyfactor user mappings for the selected logon with one click without the need to select the users on the *Users* side of the page.
   If a logon has user mappings on some servers and not others in the group (see the example, below), these will not appear in the Server Group Edit dialog, and none of these user mappings will be removed. The *Remove Shared Access for Logon* option only removes user mappings that are visible in the Server Group Access dialog.

   This option does not delete the logon from any servers (see Editing or Deleting a Logon on page 588).

7. Repeat step 6 for any other user or service accounts that you wish to unmap from this logon on the servers in this server group.

8. Click **Save**.

Example: *Server Group One* contains three Linux servers—A, B and C. Linux logons for *Anne*, *Betty* and *Dave* exist on all three servers. A Linux logon for *Chuck* exists on servers A and B but not C. In Keyfactor Command:

- Anne has acquired an SSH key using My SSH Key (see My SSH Key on page 531) and an administrator has mapped it to her Linux logon for all three servers in *Server Group One*.
- Betty has acquired an SSH key using My SSH Key and an administrator has mapped it to her Linux logon account for servers A and B but not server C.
- Chuck has acquired an SSH key using My SSH Key and an administrator has mapped it to his Linux logon account for servers A and B. No Linux account exists for Chuck on server C and no user mapping has been done for Chuck for this server.
- Dave has acquired an SSH key using My SSH Key but it has not yet been mapped to his Linux logon account for any servers.

You can see these Linux logon to Keyfactor user mappings in Figure 313: Linux Logon to Keyfactor User Mappings for Anne, Betty, Chuck and Dave.

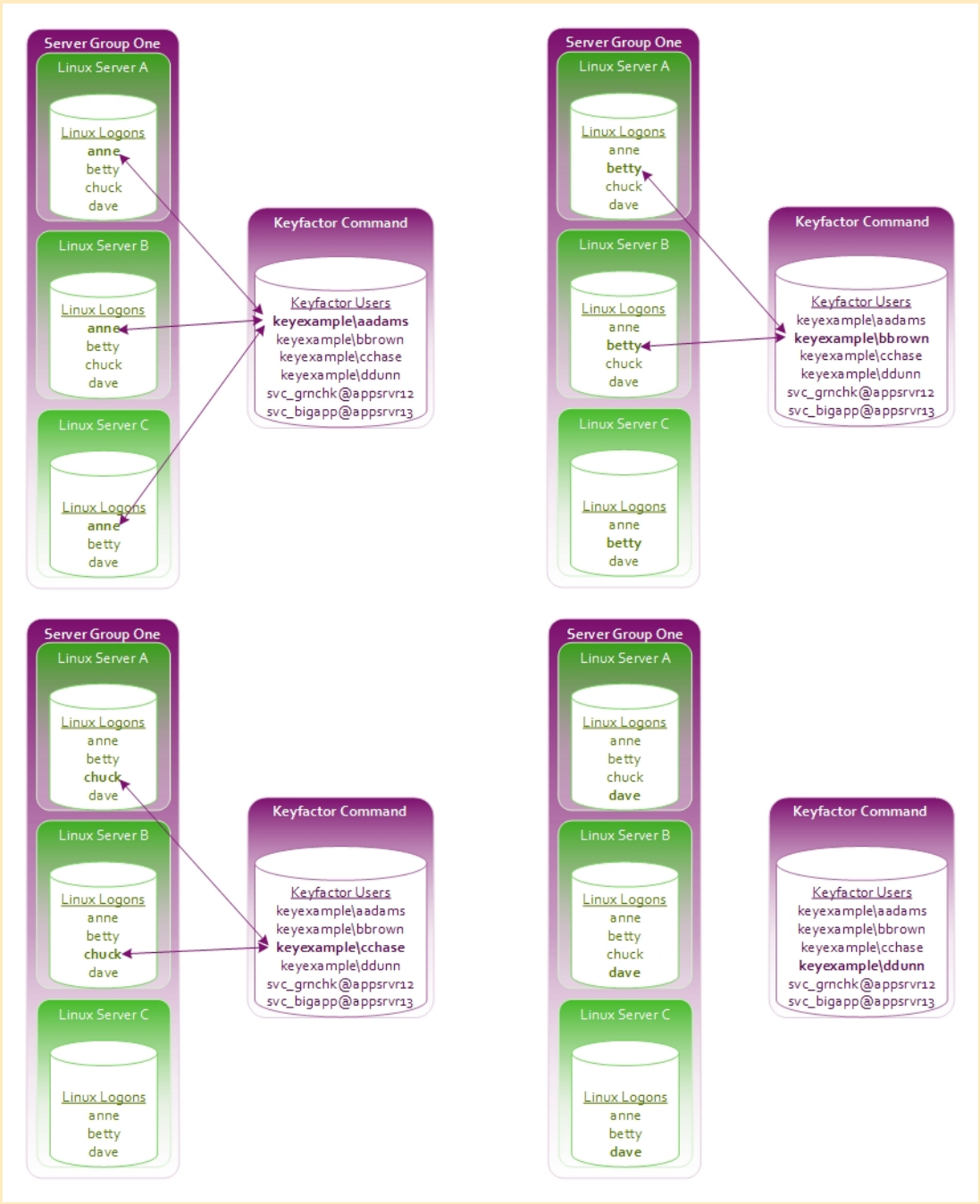


Figure 313: Linux Logon to Keyfactor User Mappings for Anne, Betty, Chuck and Dave

As a result of this logon setup and mapping configuration, when you open the Server Group Access dialog for *Server Group One* (see Figure 314: Server Group Access Editing Example), in the Logon column you will see *anne*, *betty* and *dave* but not *chuck*.

- *Chuck* is missing because he does not have a Linux logon account on server C.
- As you click on each of the users *anne*, *betty* and *dave* in the Logon column, on the right in the Users column, you will see that:
  - *Anne's* mapped user appears, but a mapped user does not appear for either *Betty* or *Dave*.
  - In *Betty's* case, this is because her Keyfactor user to Linux logon mapping does not exist for server C. Mapped users only appear if they are consistent across all Linux logons for a user.
  - In *Dave's* case, this is because he does not have a Keyfactor user to Linux logon mapping for any of the servers.
- Other shared Linux logons exist on the servers—such as root—that are not referenced in this example but are shown in Figure 314: Server Group Access Editing Example.

Tip: Logons only appear in the Linux logon column if they exist with the same spelling on all servers in the server group—*dave* does not equal *david* and will not be recognized as a Linux logon match.

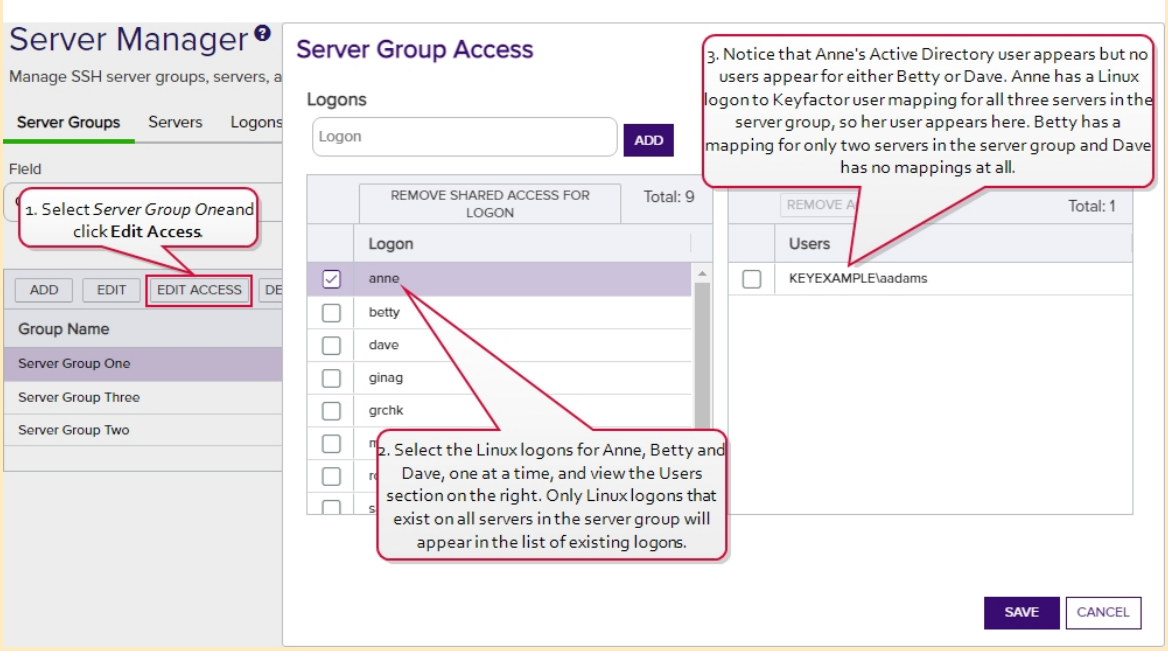


Figure 314: Server Group Access Editing Example

The administrator decides to do the following:

- On the Server Groups tab, she selects *Server Group One* and clicks **Edit Access** at the top of the grid.

- In the Server Group Access for *Server Group One*, she adds a Linux logon for *chuck* on the left and clicks **Save** without adding any user mappings on the right.
- Since *Chuck* already had Linux logon accounts on servers A and B, no changes are made on those servers. A Linux logon account is added on server C for *Chuck*.
- When the administrator opens the Server Group Access for *Server Group One* again, she sees *Chuck's* Linux logon on the left. When she clicks on *chuck*, no Users are shown on the right because *Chuck* only has Linux logon to Keyfactor user mappings for servers A and B, not for server C.

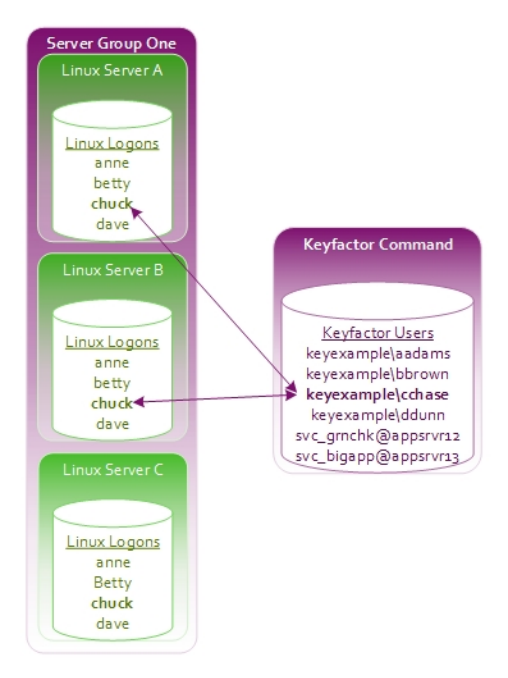


Figure 315: Concept: Add Linux Logon for Chuck on Server C

- In the Server Group Access for *Server Group One*, she selects *chuck* on the left and creates a mapping to *Chuck's* SSH key acquired through My SSH Key. This adds the key to the authorized_keys file for *Chuck* on any servers in the server group that lack the key—in this case, server C. This then completes the mappings for Linux logon the Keyfactor user for *Chuck* for the servers in this server group. *Chuck's* user will then appear in the Server Group Access dialog when *Chuck's* Logon is selected.

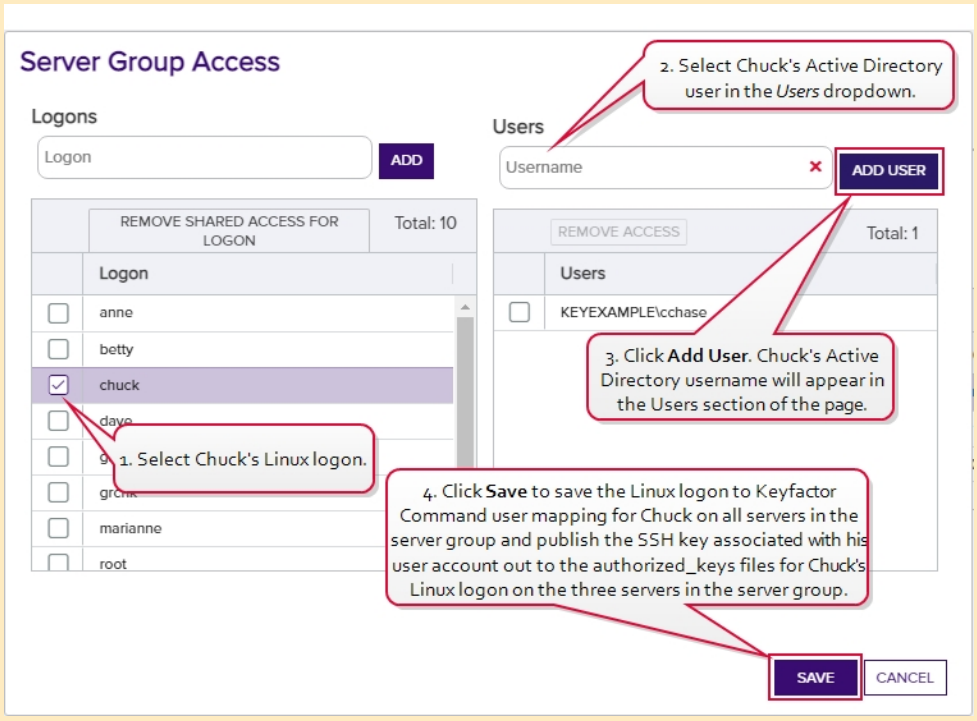


Figure 316: Server Group Access: Add Linux Logon for Chuck on Server C

- In the Server Group Access for *Server Group One*, she selects *betty* on the left and creates a mapping to *Betty's* Active Directory account, which is associated with the SSH key acquired through My SSH Key, and to a service account key for *Betty—svc_grnch-k@appsrvr12* and clicks **Save**. Since *Betty* already had Linux logon to Keyfactor user mappings for servers A and B and her SSH key was already on these servers, no changes are made to these servers. Her key acquired through My SSH Key is published out to server C and added to her authorized_keys file on that server. *Betty* had no previous mappings for the SSH service key *svc_grnchk@appsrvr12*, so this key is published out to all three servers in the server group and added to her authorized_keys files on those servers.

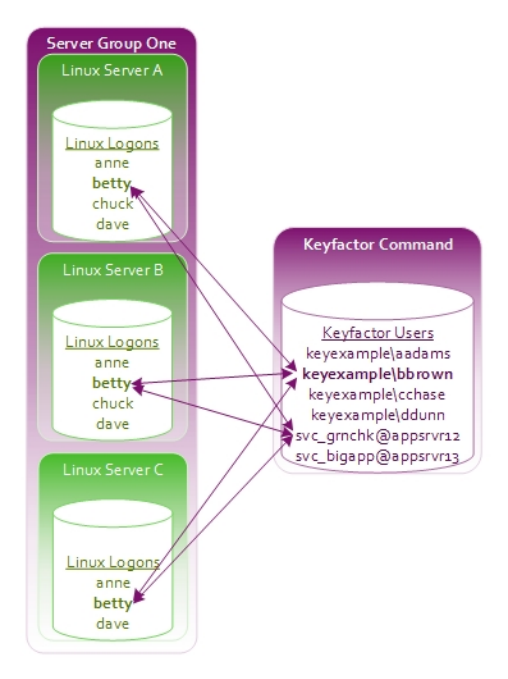


Figure 317: Add Logon to User Mapping for Betty

- At a later date, the administrator decides to remove Betty's access to the service account key. In the Server Group Access for *Server Group One*, she selects *betty* on the left and selects the service account key *svc_grnchk@appsrvr12* on the right. She clicks **Remove Access** and then **Save**. The SSH key for the *svc_grnchk@appsrvr12* service is removed from Betty's authorized_keys file on servers A, B and C. When she opens the Server Group Access dialog again and selects Betty, she sees Betty's Active Directory account, associated with the SSH key acquired through My SSH Key, but not the svc_grnchk@appsrvr12 service account.

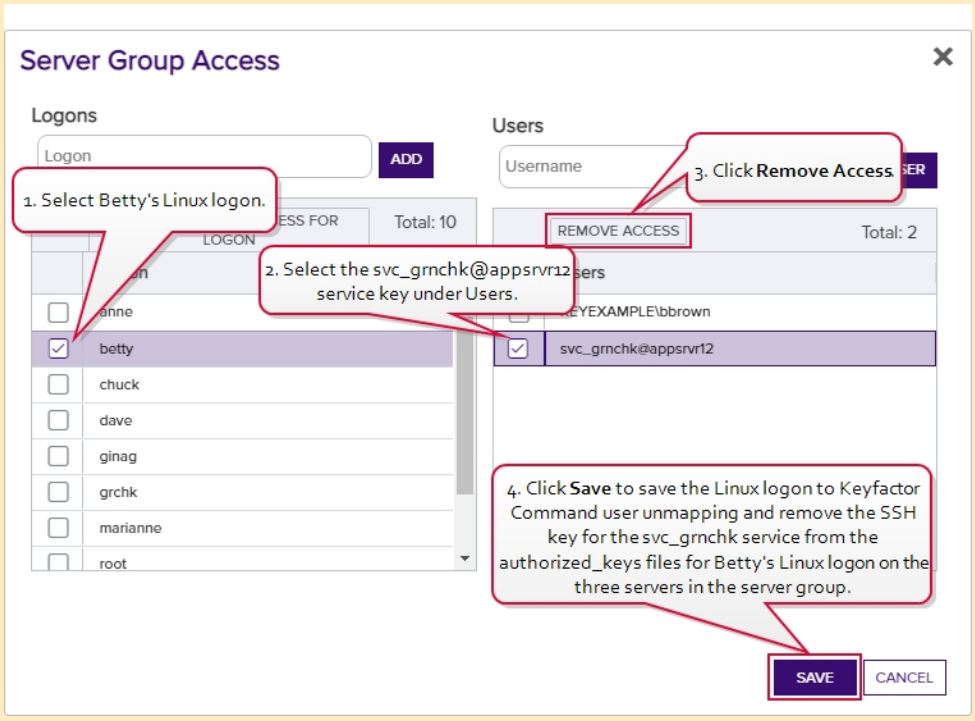


Figure 318: Remove Logon to User Mapping for Betty

- She decides to add Dave's key to the servers. On the Logons tab, she selects one of *Dave's* Linux logons on one of the servers in *Server Group One* and clicks **Edit** at the top of the grid. In the Edit Logon dialog, she changes to the Access Management tab, selects *Dave's* Active Directory account in the Users dropdown, and clicks **Add User** and **Save**. She repeats these steps for all the servers in *Server Group One* (servers A, B and C). *Dave's* SSH key acquired through My SSH Key is published out to all three servers in *Server Group One* and added to his authorized_keys files on those servers.

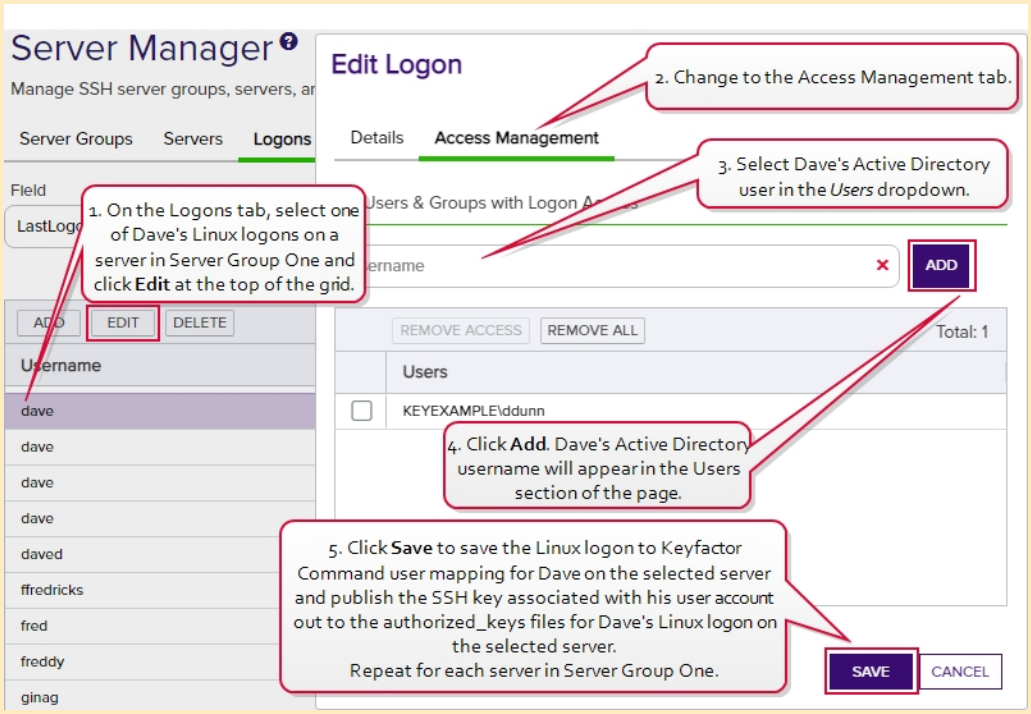


Figure 319: Add Individual Logon to User Mappings for Dave

- On the Server Groups tab, she selects *Server Group One* and clicks **Edit Access** at the top of the grid. In the Server Group Access for *Server Group One*, she selects *Dave's* Linux logon and sees that *Dave's* Active Directory account appears in the Users section on the right when it did not previously. This is because she created Linux logon to Keyfactor user mappings for *Dave* individually on the Logons tab for all the servers in *Server Group One*. If she had only done this for some of the servers in *Server Group One*, *Dave's* Active Directory user would not appear on the Server Group Access dialog.

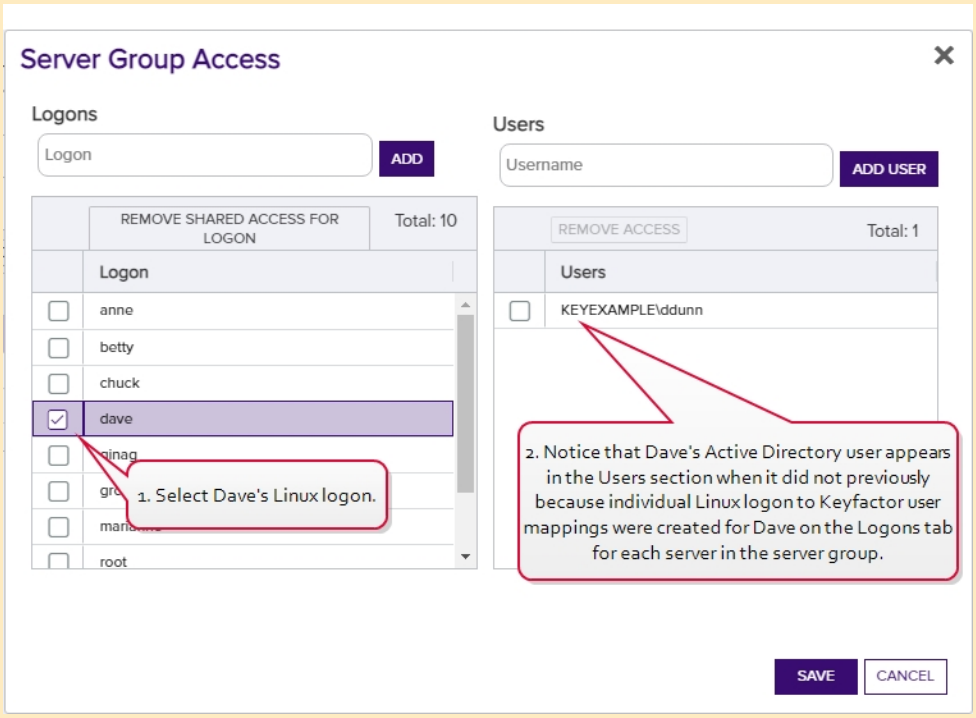


Figure 320: View Server Group Logon to User Mappings for Dave

Viewing Server Group Members

To view the servers belonging to a server group, highlight the row in the server groups grid and click **View Group Members** at the top of the grid or right-click the group in the grid and choose **View Group Members** from the right-click menu. This will take you to the Servers tab with the advanced search populated by a query for the selected server group name.

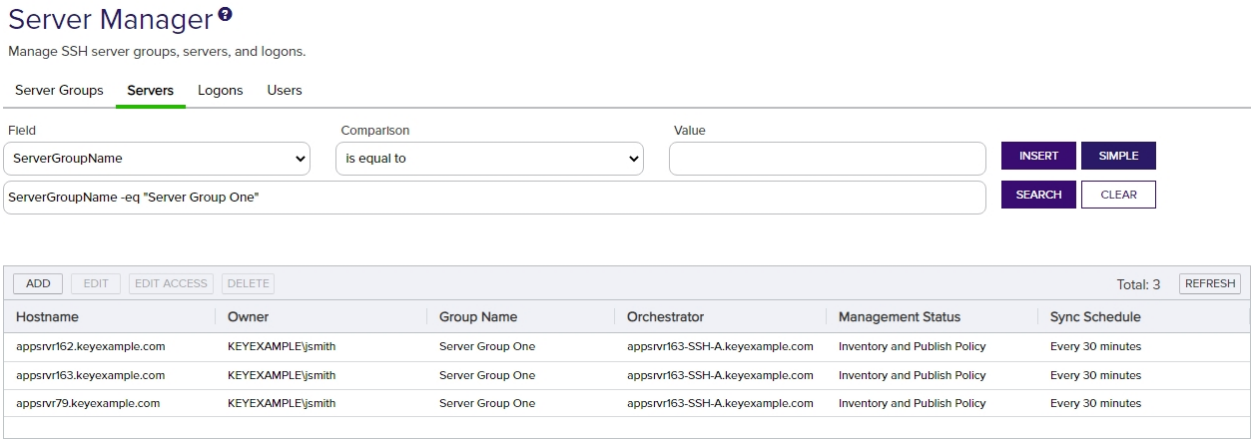


Figure 321: View Members of an SSH Server Group

Using the Server Group Search

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

GroupName

Complete or partial matches with the server group name.

OwnerName

Complete or partial matches with the Active Directory username of the user who owns the server group. The owner can only be set by a Keyfactor Command user with the SSH Enterprise Admin role.

Enforce Publish Policy

Server group is set to *enforce publish policy* yes/no.

Tip: If a specific server in a server group is not operating as expected from an inventory and policy publishing mode perspective, check the inventory and publish policy state of the individual server. The setting on the server overrides the setting on the server's group.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is greater than (-gt)
- Is greater than or equal to (-ge)

- Is less than (-lt)
- Is less than or equal to (-le)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

In addition to the options available in the query builder, three special values can be used in selected searches by typing them in directly:

SSH Servers

On the Servers tab of the Server Manager page you enter records for all the SSH servers in the environment that will be inventoried or managed with the Keyfactor Bash Orchestrator. Each SSH server added here must have either the orchestrator installed on it or have had the remote install script for the orchestrator run on it, which sets up the machine for remote control by the orchestrator. For more information about the orchestrator, see Bash Orchestrator on page 2991 in the *Keyfactor Orchestrators Installation and Configuration Guide*.

You must create at least one server group before you can add SSH servers into the Keyfactor Command Management Portal (see SSH Server Groups on page 560).

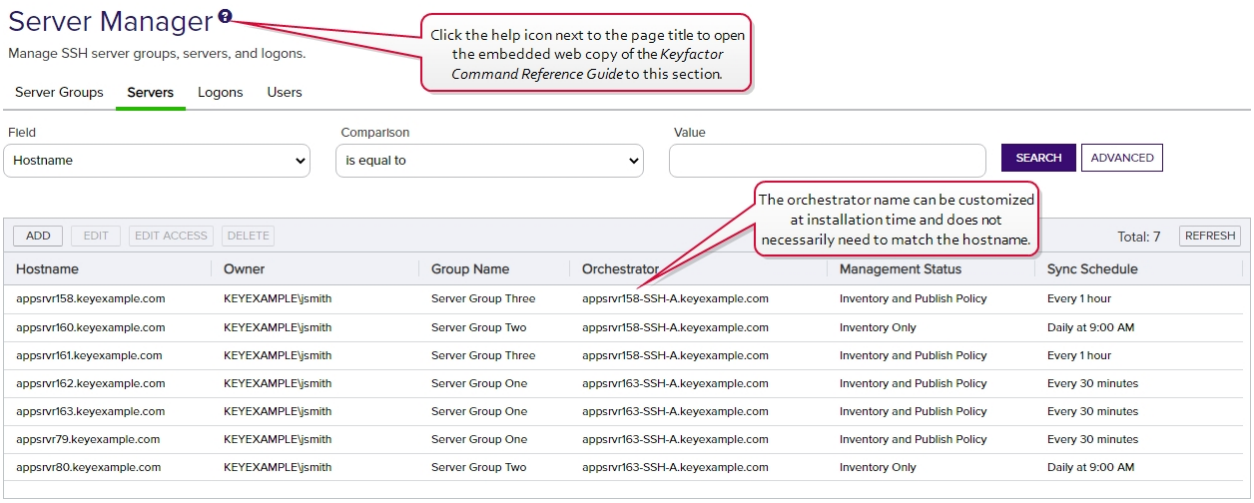


Figure 322: SSH Servers Grid

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Reference Guide* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Adding SSH Servers

Before adding a new SSH server, be sure that you have added at least one server group (see Adding Server Groups on page 561) and that your Keyfactor Bash Orchestrator has been registered and approved in Keyfactor Command (see Orchestrator Management on page 496).

To add a new SSH server:

1. In the Management Portal, browse to *SSH > Server Manager*.
2. On the Server Manager page, select the Servers tab.
3. On the Servers tab, click **Add**.

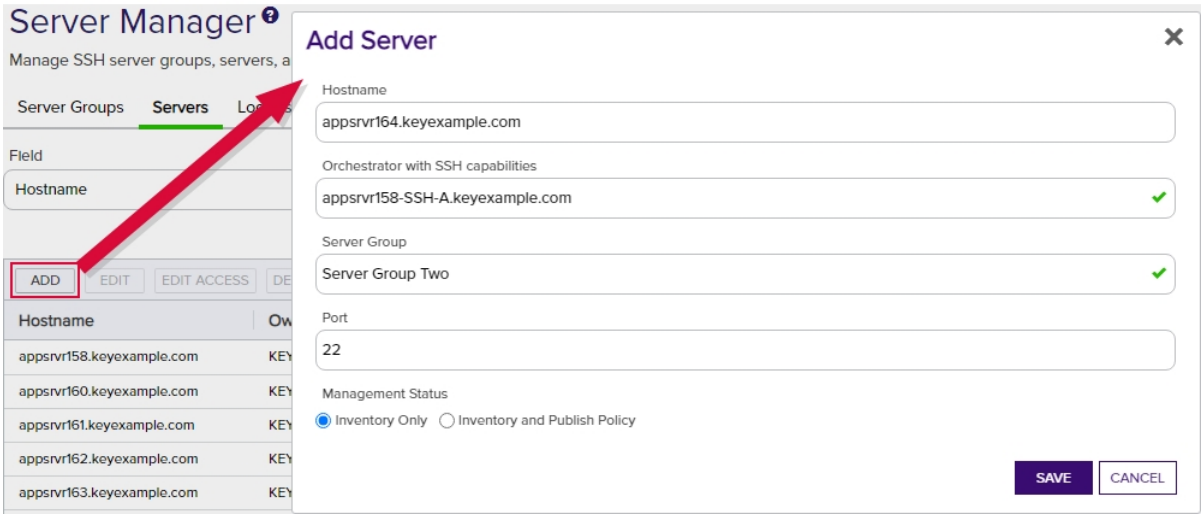


Figure 323: Add an SSH Server

4. In the Add Server dialog on the Basic tab, enter the DNS hostname for the server in the **Hostname** field. This can be either the FQDN or a short name. An IP address may be used if desired. This field is required.

   Note: The following values are **not** supported in the Hostname field:
   - 127.0.0.1
   - localhost
   - ::1

5. In the **Orchestrator** dropdown, select an approved orchestrator. This field is required.
6. In the **Server Group** dropdown, select an existing server group. This field is required.
7. In the **Port** field, either select the default SSH port of 22 or enter a custom port if an alternative port is used for SSH in your environment.
8. Select either the **Inventory Only** radio button or the **Inventory and Publish Policy** radio button (see SSH on page 525).

Tip: If the server group you selected above is configured in *inventory and publish policy* mode (with the *Enforce Publish Policy* box checked), you will not be able to save the server in *inventory only* mode.

9. Click **Save** to save the new server.

Tip: When you are first creating server records, you probably won't need to visit the Access Management tab of the server record. On this tab, you create mappings between Keyfactor Command user accounts associated with SSH keys and Linux logons in order to publish the SSH keys to the Linux servers (see SSH on page 525 and Editing or Deleting an SSH Server below).

Editing or Deleting an SSH Server

To edit a server, double-click the server, right-click the server and choose **Edit** from the right-click menu, or highlight the row in the servers grid and click **Edit** at the top of the grid.

Only two of the fields are available for editing:

- Port

  Change the SSH port set for the server, if desired.
- Management Status

  Select either the **Inventory Only** radio button or the **Inventory and Publish Policy** radio button.

  Tip: If the server group for the server is configured in *inventory and publish policy* mode (with the *Enforce Publish Policy* box checked), you will not be able to save the server in *inventory only* mode.

To delete a server, highlight the row in the servers grid and click **Delete** at the top of the grid or right-click the server in the grid and choose **Delete** from the right-click menu.

Tip: The hostname, orchestrator, and server group for a server are not editable. If you wish to change one of these, delete the record and add a fresh record for the server.

Editing Access to an SSH Server

Using the Edit Access function you create mappings between Keyfactor Command user accounts associated with SSH keys and Linux logons in order to publish the SSH public keys to the Linux servers (see SSH on page 525). You can also remove the mappings from here, which causes the SSH public keys to be removed from the Linux servers.

Before adding a logon to user mapping, be sure that you have switched the server to which you will add your mapping (or its server group) to *inventory and publish policy* mode (see Server Manager on

page 560) so that the key for the user will be published to the server. If the server is in *inventory only* mode and you add a mapping for it in Keyfactor Command, the mapping will appear in Keyfactor Command only and the key for the user will not be published out to the server.

To edit the access for a server, create a mapping between a Linux logon and a Keyfactor Command user, and publish the user's key to the SSH server:

1. In the Management Portal, browse to *SSH > Server Manager*.
2. On the Server Manager page, select the Servers tab.
3. In the Servers grid, locate the server that you wish to publish an SSH key to by mapping a Keyfactor Command user to a Linux logon on that server.
4. Right-click the server and choose **Edit Access** from the right-click menu or highlight the row in the servers grid and click **Edit Access** at the top of the grid.

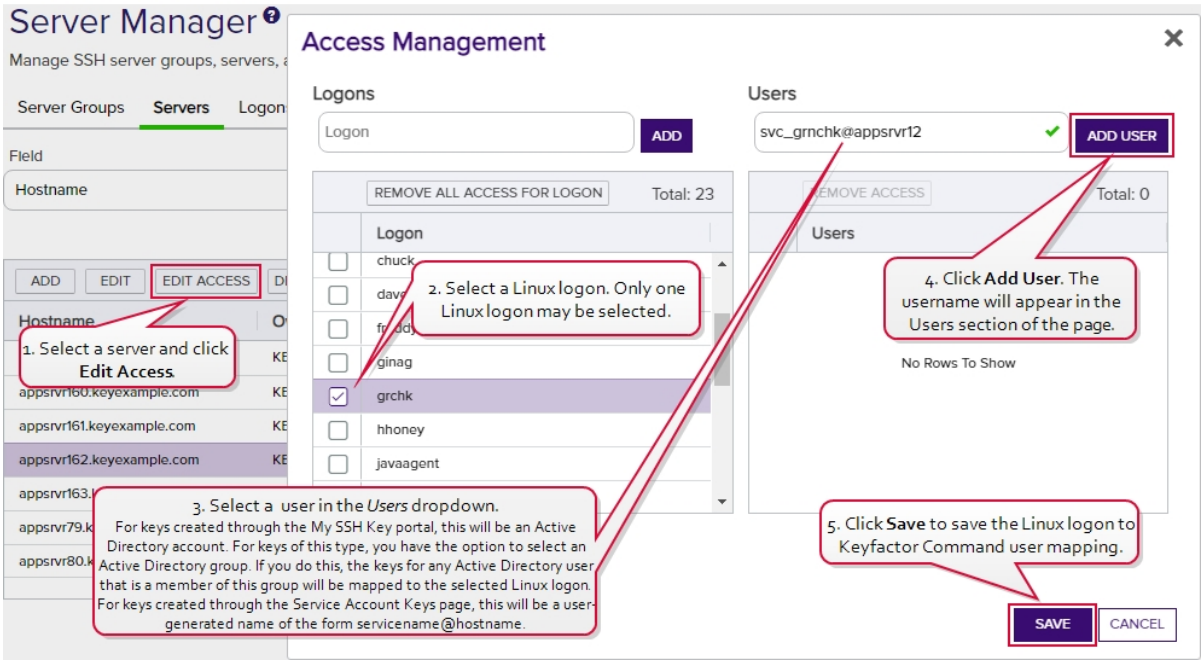


Figure 324: Edit Access for an SSH Server

5. On the Access Management page, select an existing Logon on the left side of the page. If you wish to add a new logon, enter the new logon name in the Logon field at the top of the left side of the page and click **Add Logon**. The new logon appears at the bottom of the Logon list. Click the **Logon** list title to sort the list, if desired. Select the new logon. Only one logon may be selected.

Tip: If you have enabled SSSD support for your Keyfactor Bash Orchestrator and are adding a domain user, specify the user in username@domain format. For example bbrown@keyexample.com (or, depending on SSSD configuration, such as the case-

sensitivity setting; BBROWN@keyexample.com). Note that the logon may be modified by the SSSD configuration file in ways in which Keyfactor Command cannot know about. Refer to SSH-SSSD Case Sensitivity Flag on page 791 for guidance on what to enter based on how the SSSD case sensitivity flag is configured.

6. In the Users dropdown at the top of the right side of the page, select a *user* or *service account* to associate the logon with. Only Keyfactor users that have keys stored in Keyfactor Command, that have been designated as server group owners, or AD users or groups that have been previously entered for association with a logon will appear in the dropdown. If desired, you may enter an Active Directory user or group name in this field. Using an Active Directory group to create Linux logon to Keyfactor user mappings will cause the keys stored in Keyfactor Command for any Active Directory users that are members of this group to be mapped to the selected Linux logon and published to the server on which the Linux logon exists. Any Active Directory users that are members of this group but who do not have keys stored in Keyfactor Command will not be mapped to the selected Linux logon. Click **Add User**.

   Tip: For keys created through the My SSH Key portal (see My SSH Key on page 531), a Keyfactor user is an Active Directory user account. For keys created through the Service Account Keys page (see Service Account Keys on page 542), a Keyfactor user is a user-generated service account name of the form servicename@hostname.

7. Repeat step 6 for any other user or service accounts that you wish to map to this logon on this server.

8. Click **Save**.

To remove a mapping of a Linux logon to a Keyfactor Command user for a server, removing the public key from the Linux logon's authorized_keys file:

1. In the Management Portal, browse to *SSH > Server Manager*.

2. On the Server Manager page, select the Servers tab.

3. In the Servers grid, locate the server that you wish to remove an SSH key from by unmapping a Keyfactor Command user from a Linux logon on that server.

4. Right-click the server and choose **Edit Access** from the right-click menu or highlight the row in the servers grid and click **Edit Access** at the top of the grid.

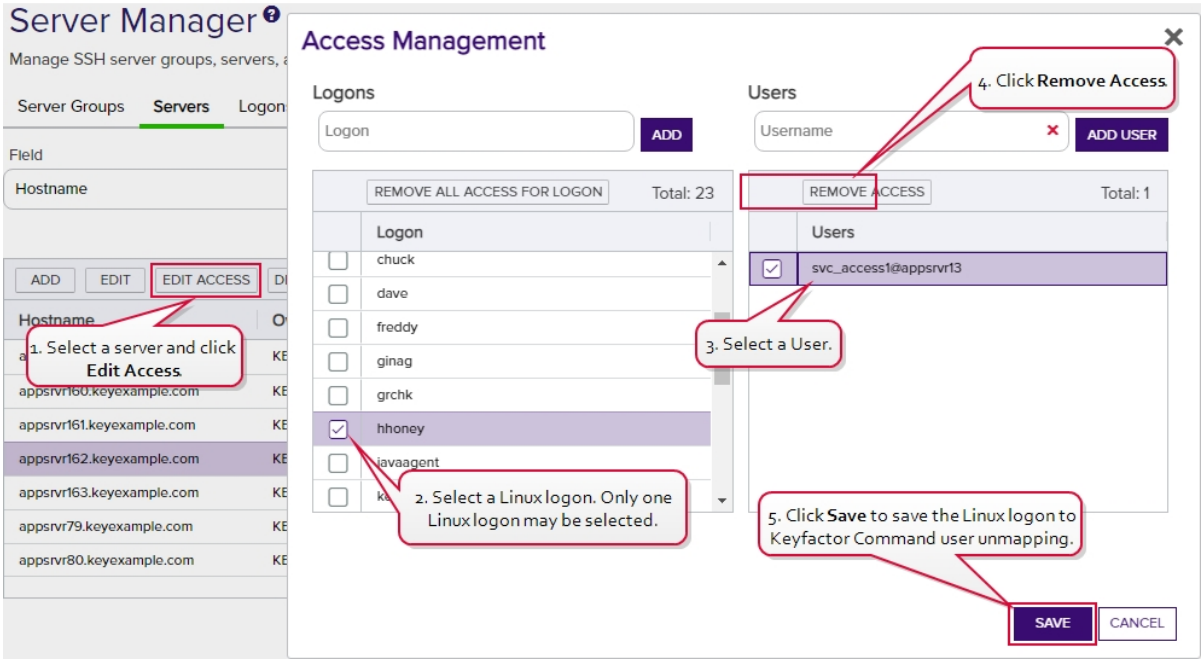


Figure 325: Edit Access for an SSH Server

5. On the Access Management page, select a Logon on the left side of the page. Only one logon may be selected.

6. In the Users section on the right side of the page, select a *user* or *service account* to unmap from the logon. Click **Remove Access** under *Users*. The Linux logon to Keyfactor user mapping for the *selected user* will be removed and the user's SSH key will be removed from the authorized_keys files of the Linux logon on the selected server.

   Tip: Clicking **Remove All Access for Logon** on the *Logons* side of the page removes *all* Linux logon to Keyfactor user mappings for the selected logon on the selected server with one click without the need to select the users on the *Users* side of the page. This option does not delete the logon from any servers (see Editing or Deleting a Logon on page 588).

7. Repeat step 6 for any other user or service accounts that you wish to unmap from this logon on this server.

8. Click **Save**.

Tip: The time it will take for changes to access mappings to appear on your Linux server will depend on the frequency of the server synchronization configured for the server group to which the server belongs (see Adding Server Groups on page 561).

Using the SSH Server Search

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

Hostname

Complete or partial matches with the hostname of the SSH server.

ServerGroupName

Complete or partial matches with the name of the server group to which the SSH servers belong.

Orchestrator

Complete or partial matches with the orchestrator controlling the SSH servers.

EnforcePublishPolicy

Server is in *inventory only* mode or *inventory and publish policy* mode.

ServerGroupOwner

Complete or partial matches with the Active Directory username of the user who owns the server group to which the server belongs. The owner can only be set by a Keyfactor Command user with the SSH Enterprise Admin role.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is greater than (-gt)
- Is greater than or equal to (-ge)

- Is less than (-lt)
- Is less than or equal to (-le)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

Logons

On the Logons tab of the Server Manager page you can view all the Linux user accounts associated with authorized_keys files containing valid SSH public keys. The logons shown here include both those discovered on SSH servers during the initial discovery phase using the orchestrator and those created in Keyfactor Command and published to the SSH servers using the orchestrator.

On this tab you can create new logons, see the number of keys associated with each logon, and create mappings between Keyfactor Command users and the logons in order to allow the orchestrator to publish new SSH keys for those users to the SSH servers (see SSH on page 525).

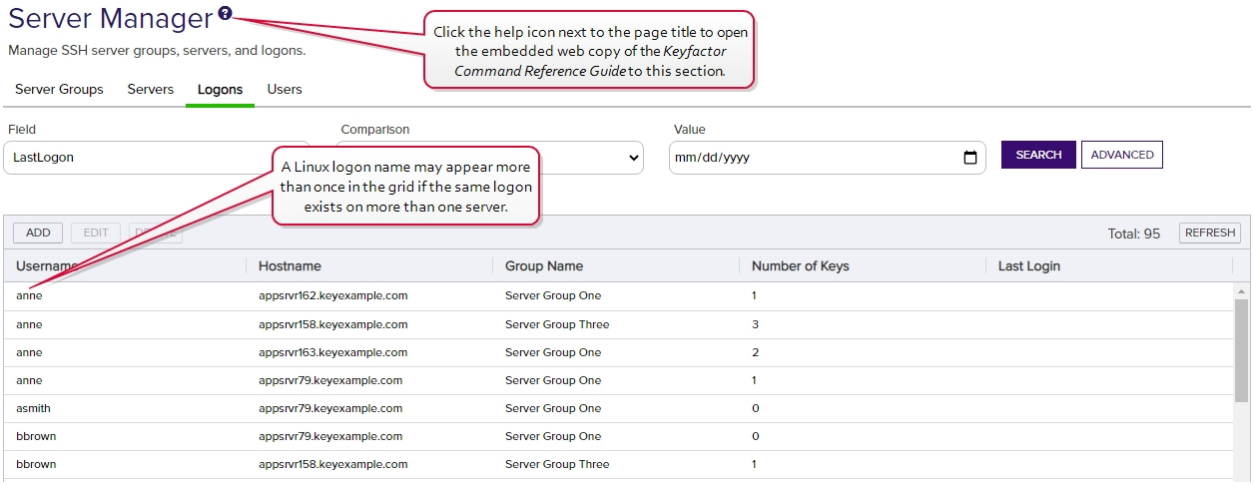


Figure 326: Linux Logons Grid

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Reference Guide* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Adding Logons

Before adding a new logon, be sure that you have switched the server to which you will add your logon (or its server group) to *inventory and publish policy* mode (see Server Manager on page 560) so that the new logon will be published to the server. If the server is in *inventory only* mode and you add a new logon for it in Keyfactor Command, the logon will appear in Keyfactor Command only and will not be published out to the server.

Tip: New logons can also be added from the access management options for server groups and servers while creating Linux logon to Keyfactor Command user mappings (see Editing

Access to an SSH Server Group on page 563 and Editing Access to an SSH Server on page 579).

To add a new logon:

1. In the Management Portal, browse to *SSH > Server Manager*.
2. On the Server Manager page, select the Logons tab.
3. On the Logons tab, click **Add**.

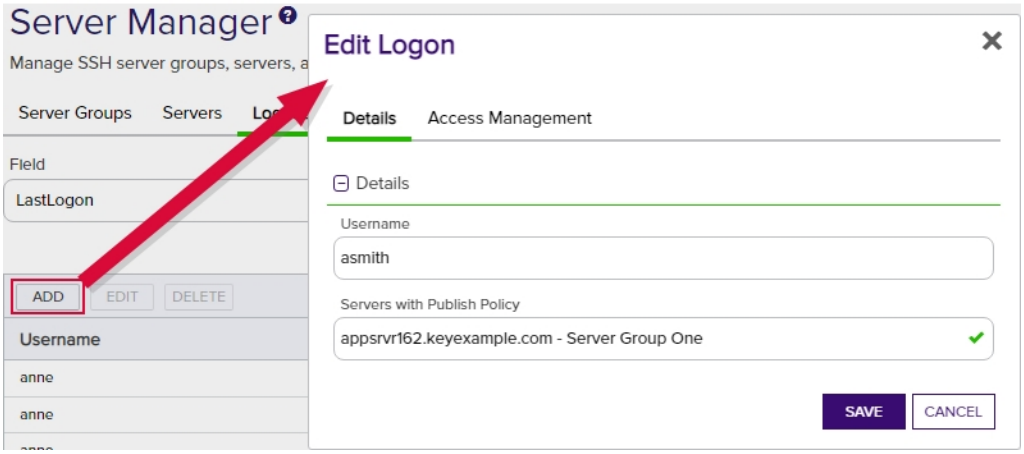


Figure 327: Add a Linux Logon—Basic Tab

4. In the Add Logon dialog on the Details tab, enter a Linux *Username* for the user.

   Tip: If you have enabled SSSD support for your Keyfactor Bash Orchestrator and are adding a domain user, specify the user in username@domain format. For example bbrown@keyexample.com (or, depending on SSSD configuration, such as the case-sensitivity setting; BBROWN@keyexample.com). Note that the logon may be modified by the SSSD configuration file in ways in which Keyfactor Command cannot know about. Refer to SSH-SSSD Case Sensitivity Flag on page 791 for guidance on what to enter based on how the SSSD case sensitivity flag is configured.

5. In the *Servers with Publish Policy* dropdown on the Details tab, select an available SSH server on which to create the logon. Only servers that are configured in *inventory and publish policy* mode (see Server Manager on page 560) will appear in this dropdown. **This field is required**.

6. On the Access Management tab in the Users & Groups with Login Access dropdown, select a *user* or *service account* to associate the logon with. Only accounts that have keys stored in Keyfactor Command or that have been designated as server group owners will appear in the dropdown. If desired, you may enter an Active Directory group name in this field. This will cause the keys stored in Keyfactor Command for any Active Directory users that are members of this

group to be mapped to the selected Linux logon and published to the server on which the Linux logon exists. Any Active Directory users that are members of this group but who do not have keys stored in Keyfactor Command will not be mapped to the selected Linux logon. Click **Add**. The Access Management tab is optional.

Tip: For keys created through the My SSH Key portal (see My SSH Key on page 531), a Keyfactor user is an Active Directory user account. For keys created through the Service Account Keys page (see Service Account Keys on page 542), a Keyfactor user is a user-generated service account name of the form servicename@hostname.

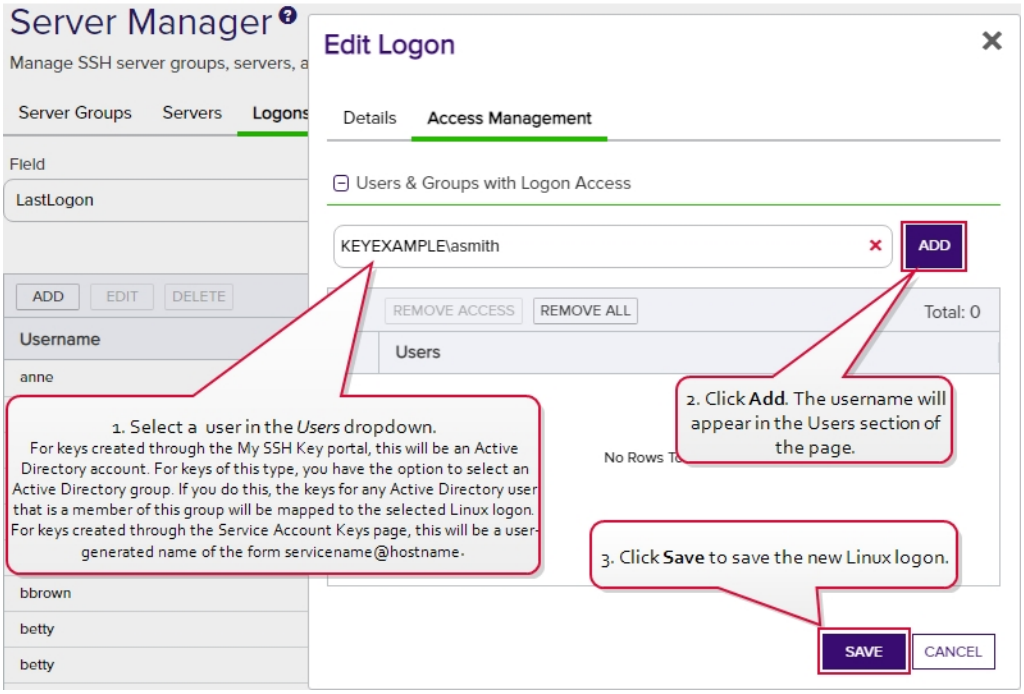


Figure 328: Add a Linux Logon—Access Management Tab

7. Click **Save** to save the new logon.

Note: When the logon is created on the Linux server, a home directory will be created for it and within this, the .ssh directory and authorized_keys file. The logon user will be made owner of the home directory and granted *rwx* permissions to it. No password is set for the user and as initially configured, the user will not be able to remotely login.

Tip: The time it will take for new logons to appear on your Linux server will depend on the frequency of the server synchronization configured for the server group to which the server belongs (see Adding Server Groups on page 561).

Editing or Deleting a Logon

On the Access Management tab of the Edit Logon dialog, you can map Keyfactor user accounts to Linux logon account to cause the SSH keys in Keyfactor Command associated with thoseKeyfactor users to be published to the authorized_keys file of the Linux user (see SSH on page 525).

To map an Keyfactor Command user to a Linux logon:

1. In the Management Portal, browse to *SSH > Server Manager*.
2. On the Server Manager page, select the Logons tab.
3. In the Logons grid locate the logon that you wish to publish an SSH key to by mapping an Active Directory account to it. Be sure to select the logon associated with the correct server, as the same logon name may appear for multiple servers.
4. Double-click the logon, right-click the logon and choose **Edit** from the right-click menu, or highlight the row in the logons grid and click **Edit** at the top of the grid.
5. On the Access Management tab in the Users & Groups with Login Access dropdown, select a *user* or *service account* to associate the logon with. Only Keyfactor users that have keys stored in Keyfactor Command, that have been designated as server group owners, or AD users or groups that have been previously entered for association with a logon will appear in the dropdown. If desired, you may enter an Active Directory user or group name in this field. Using an Active Directory group to create Linux logon to Keyfactor user mappings will cause the keys stored in Keyfactor Command for any Active Directory users that are members of this group to be mapped to the selected Linux logon and published to the server on which the Linux logon exists. Any Active Directory users that are members of this group but who do not have keys stored in Keyfactor Command will not be mapped to the selected Linux logon. Click **Add**.

   Tip: For keys created through the My SSH Key portal (see My SSH Key on page 531), a Keyfactor user is an Active Directory user account. For keys created through the Service Account Keys page (see Service Account Keys on page 542), a Keyfactor user is a user-generated service account name of the form servicename@hostname.

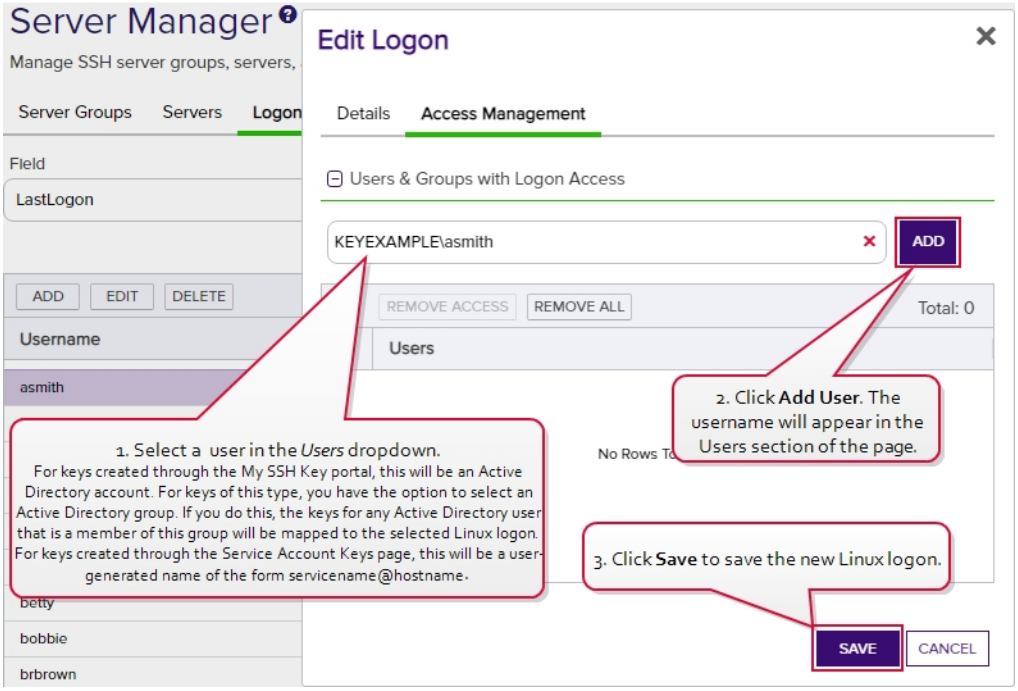


Figure 329: Edit Access for a Linux Logon

6. Click **Save** to save the access management settings.

Tip: Only the mappings of Keyfactor users to Linux logons on the Access Management tab are editable in an existing logon record. Nothing on the Details tab of the Edit Logon dialog is editable.

Note: If you opt to create Linux logon to Keyfactor user mapping using Active Directory groups, be aware that the key count values shown on the Logons grid will not reflect the keys associated with the members of the groups.

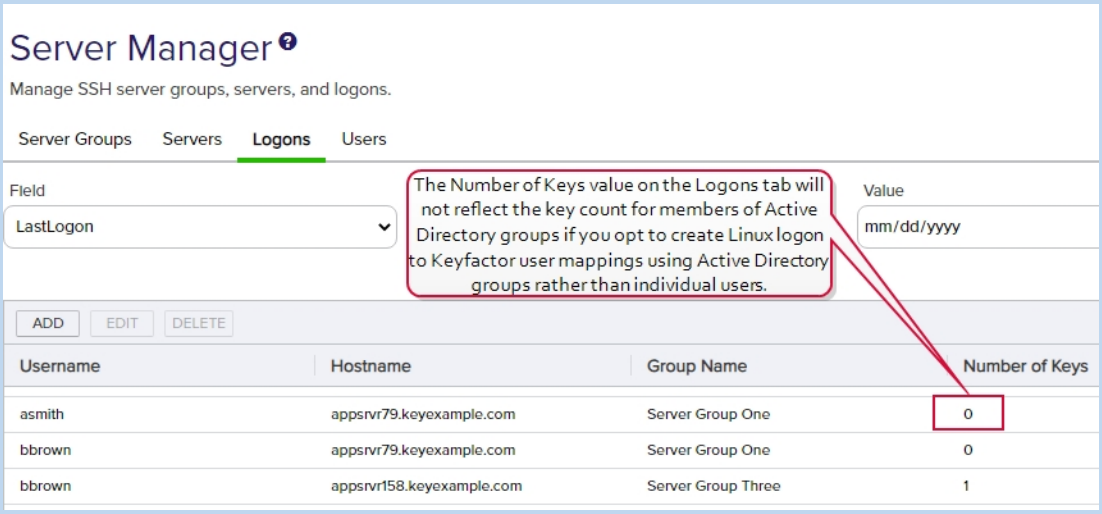


Figure 330: Creating Linux Logon to Keyfactor User Mappings Using Active Directory Groups Key Value

To delete a logon, highlight the row in the logons grid and click **Delete** at the top of the grid or right-click the logon in the grid and choose **Delete** from the right-click menu.

Note: Deleting a logon in Keyfactor Command does not delete it on the Linux server. It must be manually removed from the Linux server at the same time. If this is not done, when the next inventory of the Linux server is performed, the logon will be recreated in Keyfactor Command. This function is intended primarily to be used to clean up logons from SSH servers that have been retired.

Using the Logons Search

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

Username	Hostname
Complete or partial matches with the Linux logon	Complete or partial matches with the hostname of

name of the user account on the SSH server.

the SSH server on which the logon resides.

LastLogon

The date on which the logon was last used to login to the given hostname.

UnmanagedKeyId

The Keyfactor Command reference ID of the unmanaged key(s) associated with the logon.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

In addition to the options available in the query builder, three special values can be used in selected searches by typing them in directly:

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

> **Important:** The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

SSH Users

On the Users tab of the Server Manager page you can view all the SSH users defined in Keyfactor Command. Both *users* and *service accounts* are included. See SSH on page 525 for more

information on the difference between users and service accounts. Active Directory groups may also be included if they have previously been used to create Linux logon to Keyfactor user mappings (see Editing Access to an SSH Server on page 579). Groups appear without associated keys (since keys are associated with the member users, not the groups). Users may appear here without associated keys if the user account has been used to grant ownership on a server group but the user has not requested an SSH key pair.

On this tab you can see the keys associated with each user and create mappings between the users and Linux logons in order to allow the orchestrator to publish new SSH keys for those users to the SSH servers associated with the selected Linux logons (see SSH on page 525).

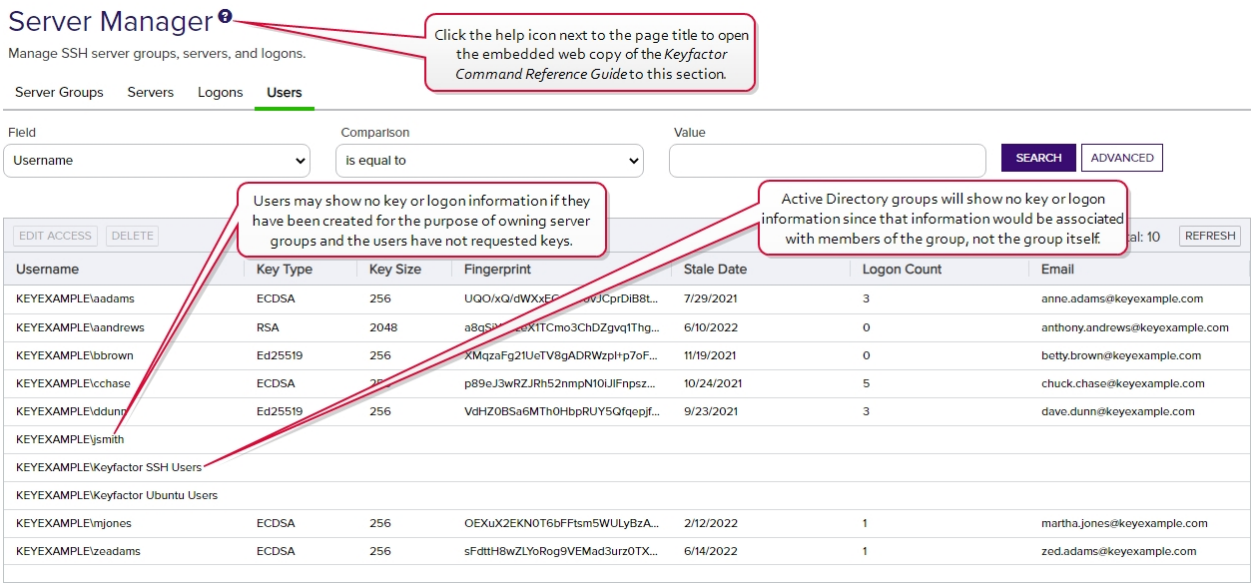


Figure 331: SSH Users Grid

Editing or Deleting an SSH User

On the Details tab of the Edit User dialog, you can view details about the user and associated key. On the Access Management tab of the Edit User dialog, you can map Keyfactor user accounts to Linux logon account to cause the SSH keys in Keyfactor Command associated with those Keyfactor users to be published to the authorized_keys file of the Linux user (see SSH on page 525).

Tip: For keys created through the My SSH Key portal (see My SSH Key on page 531), a Keyfactor user is an Active Directory user account. For keys created through the Service Account Keys page (see Service Account Keys on page 542), a Keyfactor user is a user-generated service account name of the form servicename@hostname.

To map an Keyfactor user to a Linux logon:

1. In the Management Portal, browse to *SSH > Server Manager*.
2. On the Server Manager page, select the Users tab.

3. In the Users grid locate the user whose key you wish to publish to one or more Linux logons.
4. Double-click the user, right-click the user and choose **Edit Access** from the right-click menu, or highlight the row in the users grid and click **Edit Access** at the top of the grid.
5. On the Access Management tab in the Login Access dropdown, select a logon to associate the user or service account with. A logon will appear more than once if it exists on more than one server. Be sure to select the logon on the correct server. Click **Add**.

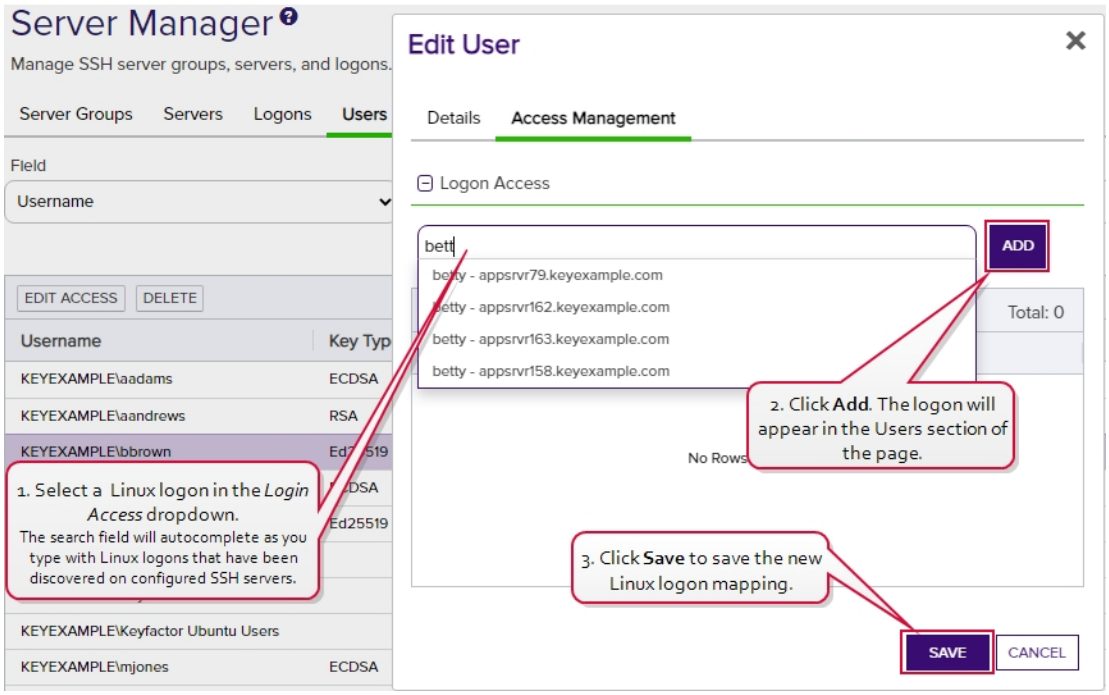


Figure 332: Edit Access for a Keyfactor User

6. Click **Save** to save the access management settings.

Tip: Only the mappings of Keyfactor users to Linux logons on the Access Management tab are editable in an existing user record. Nothing on the Details tab of the Edit Users dialog is editable.

To delete a user, highlight the row in the users grid and click **Delete** at the top of the grid or right-click the user in the grid and choose **Delete** from the right-click menu.

Using the SSH Users Search

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

Username

Complete or partial matches with the username of the user. Keyfactor users (based on Active Directory users), Active Directory groups, and service accounts are included in the grid. For Active Directory users and groups, the username is in the form DOMAIN\username. For service accounts, the username is made up of the username and client hostname entered when the service account key was created (e.g. myapp@appsrvr75). Supports the %ME% token (see Advanced Searches on the next page).

KeyType

A number of cryptographic algorithms can be used to generate SSH keys. Keyfactor Command supports RSA, Ed25519, and ECDSA. RSA keys are more universally supported, and this is the default key type when generating a new key.

KeyLength

The key size available when generating a new key depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA. The default key length is 2048.

Fingerprint

The fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.

Email

The email address of the user requesting the key. This email address is used to alert the user when the key pair is approaching the end of its lifetime (see Key Rotation Alerts on page 203).

StaleDate

The date on which the SSH key pair is considered to have reached the end of its lifetime. By default, the lifetime of an SSH key pair is 365 days (see Application Settings: SSH Tab on page 620). Supports the %TODAY% token (see Advanced Searches on the next page).

LogonCount

The number of Linux logons associated with the user.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

Important: The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

2.1.10.5 SSH Permissions

Permissions to use the SSH areas of Keyfactor Command are controlled with three security roles specific to this purpose:

- Enterprise Admin
- Server Admin
- User

Most functions in the Management Portal are available to users with the Server Admin role for SSH. The Enterprise Admin role is used to grant administrators the permission to create server groups and change the owner of a server group (see SSH Server Groups on page 560). Other than these two things, users with the Server Admin role and those with the Enterprise Admin role have the same level of access. Users with the User role (and neither of the SSH admin roles) can access only the My SSH Key page to allow them to generate an SSH key pair for their own use.

Tip: Permissions for the SSH reports and the key rotation alerts (see Key Rotation Alerts on page 203) are covered by the standard reporting and workflow permission roles, not by the specialized SSH permission roles.

Table 17: SSH Permissions Table shows the access users with each of these roles has to the SSH functions within the Management Portal.

Table 17: SSH Permissions Table

Action	SSH Enterprise Admin	SSH Server Admin	SSH User
User Key: Generate and Rotate (My SSH Key)	Yes	Yes	Yes
User Key: Download (My SSH Key)	Yes	Yes	Yes
Service Account Key: View and Search for Service Account Keys	Yes	Limited[1]	No
Service Account Key: Add	Yes	Limited[2]	No
Service Account Key: Edit	Yes	Limited[3]	No
Service Account Key: Delete	Yes	Limited[4]	No
Service Account Key: Download	Yes	Limited[5]	No
Unmanaged Keys: View and Search for Unmanaged Keys	Yes	Yes[6]	No
Unmanaged Keys: Delete	Yes	Yes[7]	No
Server Group: View and Search for Server Groups	Yes	Limited[8]	No
Server Group: Add	Yes	No	No
Server Group: Edit	Yes	Limited[9]	No

[1]Users with the Server Admin role may only view and search for service account keys that are in server groups they own.
[2]Users with the Server Admin role may only create service account keys in server groups they own.
[3]Users with the Server Admin role may only view and edit service account keys that are in server groups they own.
[4]Users with the Server Admin role may only view and delete service account keys that are in server groups they own.
[5]Users with the Server Admin role may only view and download service account keys that are in server groups they own.
[6]Users with the Server Admin role may only view and delete unmanaged keys that are in server groups they own.
[7]Users with the Server Admin role may only view and delete unmanaged keys that are in server groups they own.
[8]Users with the Server Admin role may only view and search for server groups they own.
[9]Only users with the Enterprise Admin role may change the owner of a server group. Users with the Server Admin role may change other settings when editing a server group.

Action	SSH Enterprise Admin	SSH Server Admin	SSH User
Server Group: Delete	Yes	No	No
Server Group: View Members of a Server Group	Yes	Limited[1]	No
Server Group: Edit Access (map an SSH key to a logon for a server group)	Yes	Limited[2]	No
Server: View and Search for Servers	Yes	Limited[3]	No
Server: Add	Yes[4]	Limited[5]	No
Server: Edit	Yes	Limited[6]	No
Server: Edit Access (map an SSH key to a logon on a server)	Yes	Limited[7]	No
Server: Delete	Yes	Limited[8]	No
Logon: View and Search for Logons	Yes	Limited[9]	No
Logon: Add	Yes	Limited[10]	No

[1]Users with the Server Admin role may only view the servers in server groups they own.

[2]Users with the Server Admin role may only map SSH keys from user accounts to Linux logons on servers that are in server groups they own.

[3]Users with the Server Admin role may only view and search for servers that are in server groups they own.

[4]In order to create new servers, these users must also hold the Agent Management - Read role.

[5]Users with the Server Admin role may only create new servers as members of server groups that they own. In order to create new servers, these users must also hold the Agent Management - Read role.

[6]Users with the Server Admin role may only view and edit servers that are in server groups they own.

[7]Users with the Server Admin role may only map SSH keys from user accounts to Linux logons on servers that are in server groups they own.

[8]Users with the Server Admin role may only view and delete servers that are in server groups they own.

[9]Users with the Server Admin role may only view and search for logons that are in server groups they own.

[10]Users with the Server Admin role may only create new logons on servers that are members of server groups that they own.

Action	SSH Enterprise Admin	SSH Server Admin	SSH User
Logon: Edit	Yes	Limited[1]	No
Logon: Edit Access (map an SSH key to a logon)	Yes	Limited[2]	No
Logon: Delete	Yes	Limited[3]	No
User: View and Search for Users	Yes	Limited[4]	No
User: Edit Access (map an SSH key to a logon)	Yes	Limited[5]	No
User: Delete	Yes	Limited[6]	No

2.1.11 System Settings

System Settings are accessed via the settings icon at the top right of the Management Portal.

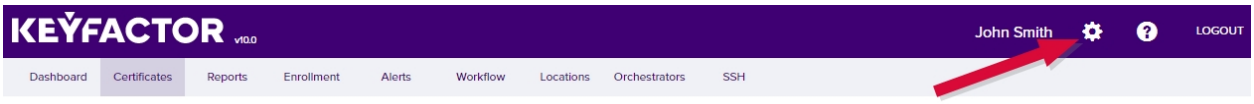


Figure 333: System Settings Icon

The options available in the System Settings section of the Management Portal are:

Application Settings

View or modify settings that control the Keyfactor Command applications.

Security Roles and Identities

Configure security roles to provide customized

Event Handler Registration

Configure built-in or custom event handlers.

Privileged Access Management

Configure PAM providers for use of Privileged Access Management (PAM) to secure certificate

---

[1]Users with the Server Admin role may only view and edit logons that are on servers in server groups they own.
[2]Users with the Server Admin role may only map SSH keys from user accounts to Linux logons on servers that are in server groups they own.
[3]Users with the Server Admin role may only view and delete logons that are on servers in server groups they own.
[4]Users with the Server Admin role may only view and search for users that are associated with logons that are in server groups they own.
[5]Users with the Server Admin role may only map SSH keys from user accounts to Linux logons on servers that are members of server groups that they own.
[6]Users with the Server Admin role may only view and delete users that are associated with logons that are in server groups they own.

levels of access to the Management Portal, configure users and/or groups and grant them access to the roles.

Certificate Store Types

Configure the types of certificate stores available for inventory, management, discovery, and reenrollment operations. This facilitates the creation of custom orchestrators to perform tasks against a wider set of certificate locations.

Certificate Metadata

Create custom metadata fields that can be used to capture additional data about certificates and report or alert based on it.

Audit Log

Display activity (e.g. creation, change, deletion) that has triggered an audit flag on a record in Keyfactor Command affecting an auditable area (e.g. Certificates, Security, Templates, Application Settings).

stores.

SMTP Configuration

Configure email.

Component Installations

View the servers on which Keyfactor Command server software is installed and the components installed on those servers.

Licensing

View or change your Keyfactor Command license.

2.1.11.1 Application Settings

Many of the settings that control the behavior of Keyfactor Command features are configurable from the **Applications Settings** on the System setting menu. Browse to *System Settings Icon* > *Application Settings*. The tables below provide a brief description of these settings.

Each tab of the Applications Settings page is organized into sections—a **General** section and additional sections based on the functionality controlled by each tab. Click the plus (+/−) next to a section to toggle expand/collapse that section.

Depending on your Keyfactor Command license, not all application settings may be applicable in your environment.

Application Settings: Console Tab

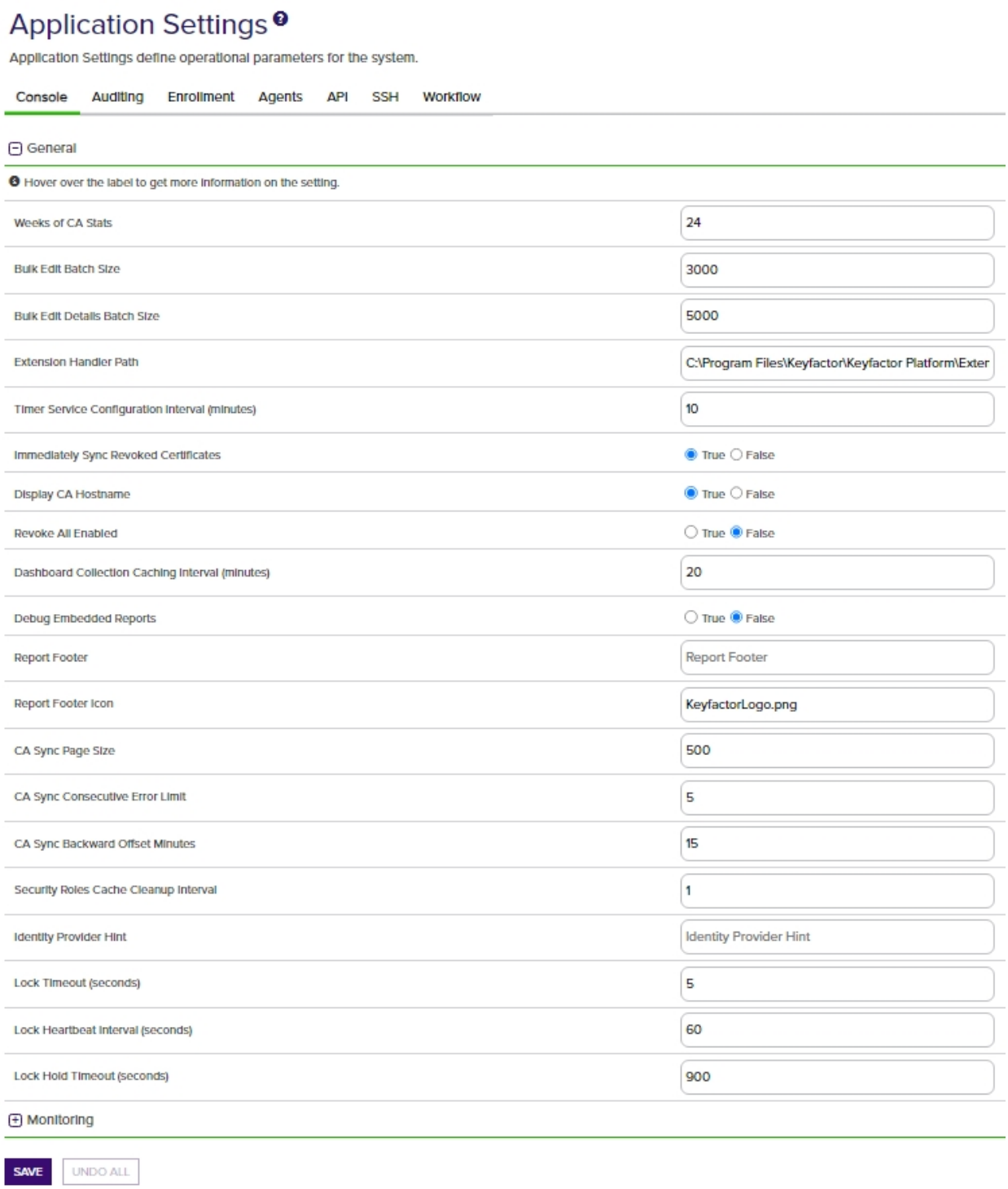


Figure 334: Console Application Settings: General

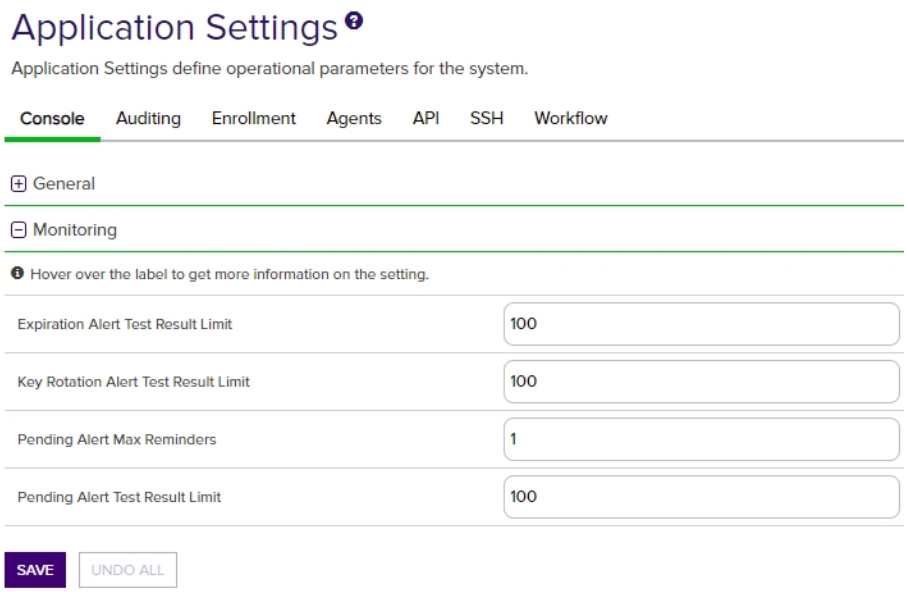
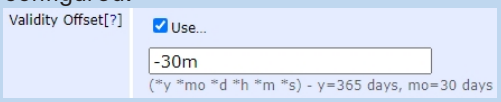


Figure 335: Console Application Settings: Monitoring

Table 18: Console Application Settings

Tab	Section	Field	Description
Console	General	Bulk Edit Details Batch Size	The number of certificates at a time that are read from the database when using the Edit All feature to edit certificate metadata. This setting can be adjusted if there are responsiveness issues when editing large numbers of certificates at once. The default value is 5000.
Console	General	Bulk Edit Batch Size	The number of certificates at a time that are saved to the database when using the Edit All feature to edit certificate metadata. This setting can be adjusted if there are responsiveness issues when editing large numbers of certificates at once. The default value is 3000.
Console	General	CA Sync Consecutive Error Limit	The number of errors a CA synchronization can encounter before the synchronization job stops (without running to completion).
Console	General	CA Sync Backward Offset	The number of minutes to offset when determining whether a certificate requested outside of

Tab	Section	Field	Description
		Minutes	Keyfactor Command should be included in an incremental synchronization. Adjusting this value can be helpful in situations of extreme clock skew or when the EJBCA *Validity Offset* setting is enabled. **Note:** For EJBCA CAs, if the certificate profile has a *Validity Offset* configured to a value greater than the value configured in the *CA Sync Backward Offset Minutes* application setting (15 minutes by default), certificates requested outside of Keyfactor Command will not be picked up on incremental scans. These certificates will only appear in Keyfactor Command on a full synchronization. The *CA Sync Backward Offset Minutes* application setting should be set to the same number of minutes as the *Validity Offset* value, if *Validity Offset* is configured.  *Figure 336: EJBCA Certificate Profile Validity Offset Greater than 15 Minutes*
Console	General	CA Sync Page Size	The number of records at a time that are read from the CA during a CA synchronization job. The default value is 500. **Note:** This setting applies only to EJBCA CAs.
Console	General	Dashboard Collection Caching Interval (minutes)	The number of minutes before data for the Collections dashboard panel is refreshed. The default value is 20.
Console	General	Weeks of CA Stats	The number of weeks of CA data to include in the dashboard graphs. The default value is 24.
Console	General	Debug Embedded	If set to **True**, causes an *Enable Debug* tickbox to

Tab	Section	Field	Description
		Reports	appear on the parameters page for reports you access and run from the Navigator (reports on the Reports menu dropdown of the Management Portal). This option does not appear for reports generated from the Report Manager grid. When enabled it allows the reports to output debug level information when they run. If set to **False**, does not display the *Enable Debug* option. The default value is False. **Tip:** When the debugging option is enabled, a small debug icon ( ) appears at the bottom of reports that generate successfully. You can click on it to see information about the report.
Console	General	Display CA Hostname	If set to **True**, causes both the CA's FQDN and logical name (e.g. ca2.keyexample.com\Corp Issuing CA Two) to display in the CA fields on the Certificate Authority, Certificate Requests and API Applications pages of the Management Portal. If set to **False**, only the CA's logical name (e.g. Corp Issuing CA Two) displays on these pages. The default value is True.
Console	General	Extension Handler Path	The path to the location on the Keyfactor Command server where the event handler .dll files are stored. By default this is: `C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\.`
Console	General	Immediately Sync Revoked Certificates	If set to **True**, causes certificates to immediately sync to Keyfactor Command upon revocation rather than waiting for the next scheduled synchronization cycle. The default value is True.
Console	General	Report Footer	A string that appears at the bottom of Logi-based reports either generated from the Management Portal or generated with the Report Manager in PDF format. The report footer appears only at the very end of the report, not at the foot of every page

Tab	Section	Field	Description
			in the report.
Console	General	Report Footer Icon	The file name of an image to be used at the bottom of each page of exported and scheduled PDF reports. You can use this to replace the Keyfactor logo with a custom image on your reports. The image is auto set to a height of 30px. This image should be placed in the **_SupportFiles** folder under the Logi folder (located at *C:\Program Files\Keyfactor\Keyfactor Platform\Logi* by default).
Console	General	Revoke All Enabled	If set to **True**, causes the Revoke All button to appear at the top of certificate search and collection grids to allow users with appropriate permissions to revoke all certificates shown in the grid or included in the certificate collection. If set to **False**, hides the Revoke All button and disables the POST /Certificates/RevokeAll API endpoint. The default value is False for new installations of Keyfactor Command beginning with release 10.4.
Console	General	Timer Service Configuration Interval (minutes)	The number of minutes between checks by the master scheduling service for changes to the synchronization schedules. Any changes made to this value will not be applied until the Keyfactor Command service is restarted. The default value is 10.
Console	General	Lock Timeout (seconds)	The amount of time to attempt to acquire a lock ensuring that only one timer service job runs at a time across multiple servers. Default is 5 seconds.
Console	General	Lock Heartbeat Interval (seconds)	How often to update the lock to keep it alive while running a long running timer service job. Default is 60 seconds.
Console	General	Lock Hold Timeout (seconds)	How long to wait after the last successful heartbeat interval before the lock is considered to be lost and can be acquired by another machine. Default is 900 seconds.
Console	Monitoring	Expiration Alert Test Result Limit	The maximum number of expiration alert emails that will be sent when an expiration alert test is run from

Tab	Section	Field	Description
			within the Management Portal. If the number set here is exceeded during a test, emails will not be sent, but a portion of the alerts will be visible on the expiration alerts test page (see Testing Expiration Alerts on page 172). The default value is 100.
Console	Monitoring	Key Rotation Alert Test Result Limit	The maximum number of key rotation alert emails that will be sent when a key rotation alert test is run from within the Management Portal. If the number set here is exceeded during a test, emails will not be sent, but a portion of the alerts will be visible on the key rotation alerts test page (see Testing Key Rotation Alerts on page 207). The default value is 100.
Console	Monitoring	Pending Alert Test Result Limit	The maximum number of pending alert emails that will be sent when a pending alert test is run from within the Management Portal. If the number set here is exceeded during a test, emails will not be sent, but a portion of the alerts will be visible on the pending alerts test page (see Testing Pending Request Alerts on page 184). The default value is 100.
Console	Monitoring	Pending Alerts Max Reminders	The maximum number of pending alert emails that will be sent for a given pending certificate. Every time a pending alert task is run, an email will be sent for a given pending certificate until the limit is reached. It is recommended that the number is kept at 5 or less. The default value is 1.

Application Settings: Auditing Tab

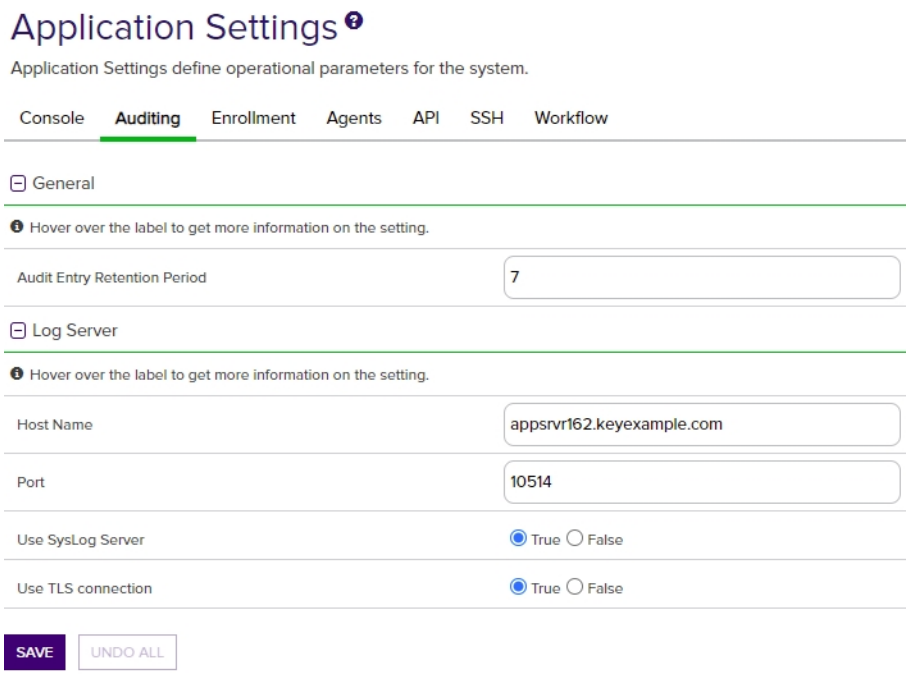


Figure 337: Audit Log Application Settings

Table 19: Audit Log Application Settings

Tab	Section	Field	Description
Auditing	General	Audit Entry Retention Period	The number of years to retain the audit log entry details. The default value is 7. **Note:** The audit log cleanup job runs once daily and removes any audit log entries older than the time specified in the retention parameter except those in the following protected categories: • Security • CertificateCollections • ApplicationSettings • SecurityIdentities • SecurityRoles Audit logs belonging to protected categories

Tab	Section	Field	Description
			are retained indefinitely and cannot be deleted. To retain all audit log entries indefinitely, disable the job. See Keyfactor Command Service Job Settings on page 778.
Auditing	Log Server	Host Name	The host name of the centralized logging server to receive the Keyfactor Command audit log entries.
Auditing	Log Server	Port	The port to connect to the centralized logging server. The default port (configurable during install) is 514.
Auditing	Log Server	Use SysLog Server	If set to **True**, enables sending audit log details to a centralized logging server. See Audit Log Output to a Centralized Logging Solution on page 805.
Auditing	Log Server	Use TLS Connection	If set to **True**, enables sending audit log details to a centralized logging server over a TLS connection. See Audit Log Output to a Centralized Logging Solution on page 805.

Application Settings: Enrollment Tab

Note: Regular expressions for enrollment that were previously configured under application settings are now configured on the templates page (see Regular Expressions on page 402).

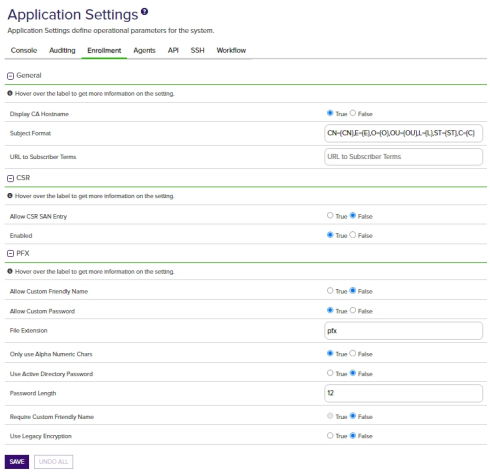


Figure 338: Enrollment Application Settings

Table 20: Enrollment Application Settings

Tab	Section	Field	Description
Enrollment	General	Display CA Hostname	If set to **True**, causes both the CA's FQDN and logical name (e.g. ca2.keyexample.com\Corp Issuing CA Two) to display in the CA dropdowns in the Keyfactor Command Management Portal interfaces. If set to **False**, only the CA's logical name (e.g. Corp Issuing CA Two) displays in these dropdowns. The default value is True.
Enrollment	General	Subject Format	The format of the subject field that will be created for the certificates requested through the Keyfactor Command Management Portal if the template used for enrollment is set to supply in request. For example: `CN={CN},E={E},O=Key Example\, Inc.,OU={OU},L=Chicago,ST=IL,C=US` The data in the subject format takes precedence over any data entered during PFX enrollment or supplied by enrollment defaults (see Enrollment Defaults Tab on page 398). For example, with the above subject format, the organization for certificates generated through PFX enrollment will always be *Key Example, Inc.* regardless of what is shown on the PFX enrollment page during enrollment. This setting applies to CSRs generated using the CSR generation method in the Keyfactor Command Management Portal and CSR and PFX enrollments done in the Keyfactor Command Management Portal. Data from the default subject *does not* display on the CSR or PFX enrollment page. To define defaults that will display in the PFX enrollment form (and can be modified by users), use enrollment defaults (see Enrollment Defaults Tab on page 398). **Note:** Backslashes are required before any commas embedded within values in the subject field (e.g. O=Key Example\, Inc.). Quotation marks should not be used in the strings in the fields except in the case where these are part of the desired subject value, as

Tab	Section	Field	Description
			they are processed as literal values. **Tip:** The default subject format *does not* apply to enrollments done using the Keyfactor API.
Enrollment	General	URL to Subscriber Terms	The URL for a web page providing terms and conditions to which a user must agree before being allowed to enroll for a certificate if the CA setting of *Require Subscriber Terms* is enabled.
Enrollment	CSR	Allow CSR SAN Entry	If set to **True**, enables the section of the CSR enrollment page that allows for entry of custom subject alternative names (SANs). The default value is False.
Enrollment	CSR	Enabled	If set to **True**, enables administrative CSR enrollment. The default value is True.
Enrollment	PFX	Allow Custom Friendly Name	If set to **True**, enables the section of the PFX enrollment page that allows for entry of a custom friendly name for the certificate. The default value is False.
Enrollment	PFX	Allow Custom Password	If set to **True**, enables the section of the PFX enrollment (see PFX Enrollment on page 146) and certificate download (see Download on page 56) pages that allow for entry of a custom password for the certificate file. The default value is False.
Enrollment	PFX	Enabled	If set to **True**, enables administrative PFX enrollment. The default value is True.
Enrollment	PFX	File Extension	The file extension that will be given to the certificate files. Typical extensions are PFX or P12. The default value is PFX.
Enrollment	PFX	Only use Alpha Numeric Chars	If set to **True**, the one-time password generated to encrypt the PFX file acquired through the Keyfactor Command Management Portal (if the user's Active Directory password is not used) will contain just numbers and letters. If set to **False**, the password will contain numbers, letters and special characters. This setting is ignored if PFX Use Active Directory Pass-

Tab	Section	Field	Description
			word is set to **True**. The default value is True.
Enrollment	PFX	Use Active Directory Password	If set to **True**, uses the user's Active Directory password to encrypt the PFX file containing the certificate acquired through the Keyfactor Command Management Portal and its private key. If set to **False**, generates a one-time password to encrypt the PFX file. The default value is False. **Important:** If you change this setting in the application settings you must also change the authentication method configured on the IIS virtual application *KeyfactorPortal* through the IIS Manager. If you set this option to *True*, you should configure only Basic Authentication in IIS. If you set this option to *False*, you may configure either only Windows Authentication or both Basic Authentication and Windows Authentication (the default) in IIS. This is because when you authenticate to the Management Portal using integrated Windows authentication (Kerberos), Keyfactor Command does not have access to your credentials to apply your password to the PFX file.
Enrollment	PFX	Password Length	The number of characters in the one-time auto-generated password, or the required number of characters in the custom password, to encrypt the PFX file acquired through the Keyfactor Command Management Portal The minimum number is 8. The default value is 12. This value will be displayed on the PFX enrollment and certificate download pages - password section (if *Allow Custom Password on the previous page* is set to true). **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative

Tab	Section	Field	Description
			access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.
Enrollment	PFX	Require Custom Friendly Name	If set to **True**, requires the user to enter a custom friendly name for the certificate. The default value is False. **Note:** If *Require Custom Friendly Name* is enabled, *Allow Custom Friendly Name* must also be enabled to place the custom friendly name field on the PFX enrollment page.
Enrollment	PFX	Enable Legacy Encryption	If set to **True**, the historical algorithm set (3DES/SHA1/RC2) is used for PFX enrollments. If set to **False**, the newer algorithm set provided by Windows (AES256/SHA256/AES256) is used instead. The default value is False. **Important:** This must be set to True if you plan to install the resulting PFX file on a server running Windows Server 2016.

Application Settings: Agents Tab

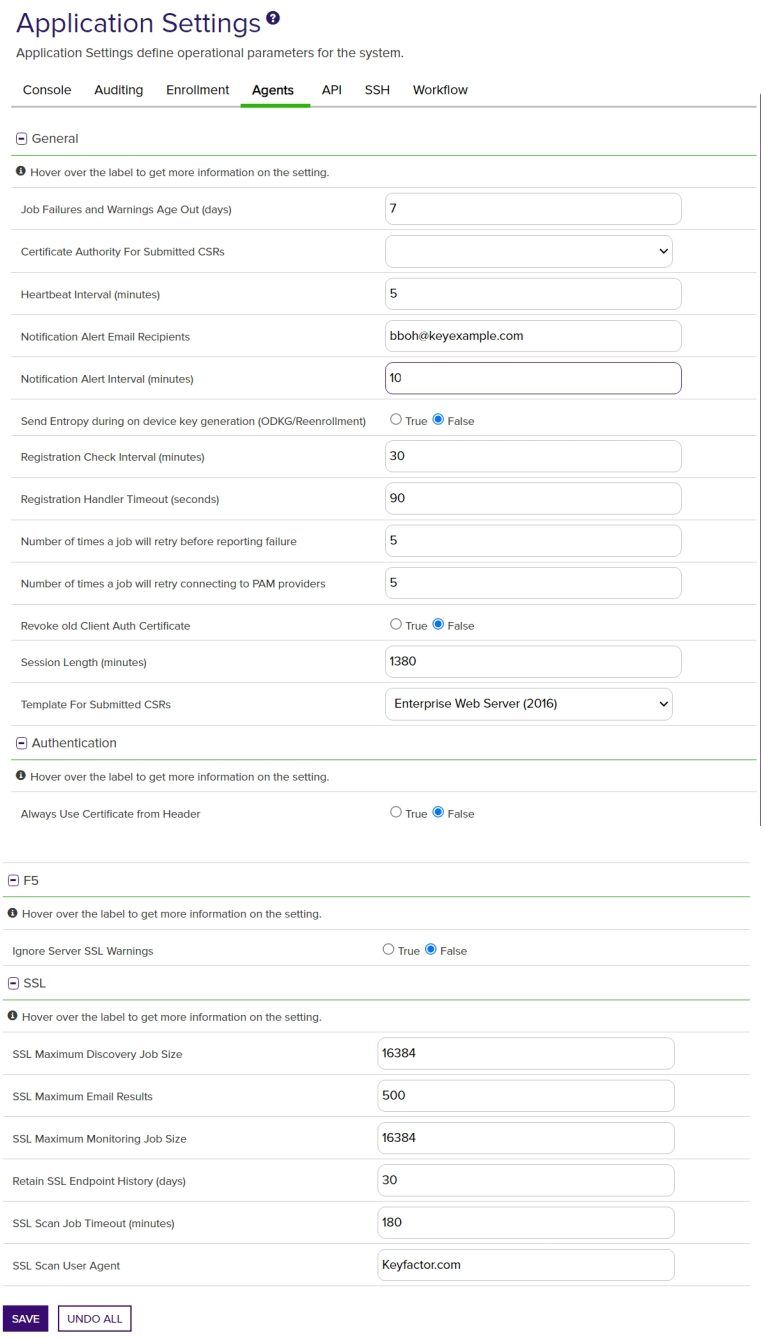


Figure 339: Agents Application Settings

Table 21: Agents Application Settings

Tab	Section	Field	Description
Agents	General	Job Failures and Warnings Age Out (days)	The number of days orchestrator job failures and warnings should be included in the count of failures on the orchestrator job history tab. The default value is 7.
Agents	General	Certificate Authority For Submitted CSRs	The certificate authority used for reenrollment requests made from the Certificate Stores page. See Certificate Store Reenrollment on page 425.
Agents	General	Heartbeat Interval (minutes)	The frequency, in minutes, with which an orchestrator (e.g. Keyfactor Universal Orchestrator, Keyfactor Java Agent or Keyfactor Mac Auto-Enrollment Agent) should query the Keyfactor Command orchestrator server for a status on the accuracy of its jobs list. The default value is 5.
Agents	General	Notification Alert Email Recipients	The email address(es) to receive notification.
Agents	General	Notification Alert Interval (minutes)	The timer service has a job that runs based on this application setting. If an orchestrator has not checked in between job runs, an email alert is sent to the configured recipients stating which orchestrator has not been seen.
Agents	General	Send Entropy during on device key generation (ODKG/Reenrollment)	Whether the configure call returns the property *Entropy* containing 2048 bytes. This property is optional via this app setting. The default is false on upgrades and new installs.
Agents	General	Registration Check Interval (minutes)	The frequency, in minutes, with which an orchestrator should check with the Keyfactor Command server to see if it has been approved as an orchestrator. The default value is 30.
Agents	General	Registration Handler	The maximum number of seconds an

Tab	Section	Field	Description
		Timeout (seconds)	auto-registration handler is allowed to attempt to run before being halted and declared to be deferred. The default value is 90 for more recently installed systems. Keyfactor recommends using a value of at least 60 seconds.
Agents	General	Number of times a job will retry before reporting failure	The number of times an orchestrator job will attempt to retry running if it encounters an error before failing. The default value is 5.
Agents	General	Revoke old Client Auth Certificate	If set to **True**, revokes the previous certificate used for orchestrator client certificate authentication after the certificate has successfully been renewed using the client certificate authentication renewal extension. The default value is True.
Agents	General	Session Length (minutes)	The frequency, in minutes, with which an orchestrator renews its session with the Keyfactor Command server and obtains a new session token in the absence of any other reason for the orchestrator to renew the session token. The session token is also renewed when an orchestrator job changes (e.g. an inventory schedule changes, a certificate is scheduled for addition to a certificate store, or a certificate is scheduled for removal from a store) or the orchestrator is restarted. The default value is 1380.
Agents	General	Template For Submitted CSRs	The template used for reenrollment requests made from the Certificate Stores page. See Certificate Store Reenrollment on page 425. The template selected for this value must be available for enrollment against the CA listed in the Certificate Authority For Submitted CSRs setting.

Tab	Section	Field	Description
Agents	Authentication	Always Use Certificate from Header	If set to **True**, the orchestrator will be authenticated using the client certificate provided in the header from the orchestrator rather than client certificate used to make the connection to Keyfactor Command. This is useful in configurations where one certificate is used to authenticate the orchestrator to a proxy and a second certificate is used to authenticate the proxy to Keyfactor Command. The original certificate from the orchestrator can be preserved in the header to present to Keyfactor Command for authentication. The default value is False.
Agents	F5	Ignore Server SSL Warnings	If set to **True**, the orchestrator will connect to the F5 device using SSL even if it detects a problem with the certificate on the F5 device (e.g. it doesn't trust the issuer of the certificate because the certificate is self-signed). This option applies only to the F5 methods based on the F5 SOAP API (see Certificate Stores on page 408). The F5 methods based on the F5 iControl REST API automatically ignore SSL warnings without the need to set this option. The default value is False.
Agents	SSL	SSL Maximum Discovery Job Size	The maximum number of endpoints for scanning that will be assigned to any one orchestrator for a given discovery scan job part. Together with the *SSL Scan Job Timeout* setting, this can be used to fine tune the running of SSL discovery scan jobs. The default value is 16,384. **Note:** A change made to this setting takes effect with the next discovery scan job. It does not affect currently running jobs.

Tab	Section	Field	Description
Agents	SSL	SSL Maximum Email Results	The maximum number of results to display in an SSL monitoring results email message table of certificates that have expired or are expiring shortly. The default value is 500.
Agents	SSL	SSL Maximum Monitoring Job Size	The maximum number of endpoints for scanning that will be assigned to any one orchestrator for a given monitoring scan job part. Together with the *SSL Scan Job Timeout* setting, this can be used to fine tune the running of SSL monitoring scan jobs. The default value is 16,384. **Note:** A change made to this setting takes effect with the next monitoring scan job. It does not affect currently running jobs.
Agents	SSL	Retain SSL Endpoint History (days)	The number of days old an endpoint history record must be before it is available for deletion by the endpoint history cleanup process. Endpoint history records older than this will be retained if they are the last records for the given endpoint. Both the last discovery and last monitoring records will be retained regardless of age. The default value is 30.
Agents	SSL	SSL Scan Job Timeout (minutes)	The maximum number of minutes any one orchestrator is allowed to attempt to run an SSL scan job before the job for that orchestrator is abandoned and given to the next orchestrator in the orchestrator pool to run (if applicable). The default value is 180. **Note:** A change made to this setting takes effect immediately. It applies to currently running jobs as well as future jobs.

Tab	Section	Field	Description
Agents	SSL	SSL Scan User Agent	Defines what is sent to endpoints when Request Robots.txt is enabled on a SSL Network.

Application Settings: API Tab

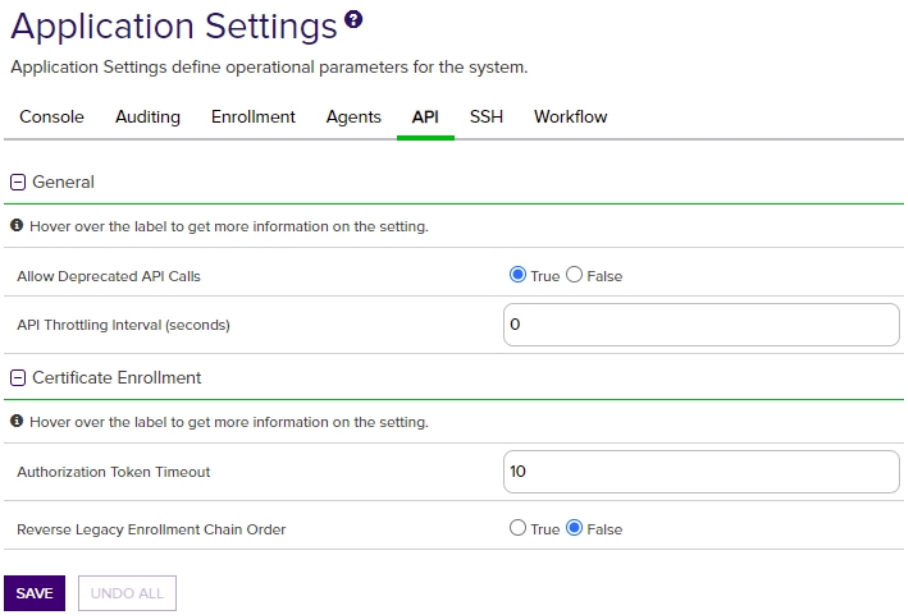


Figure 340: API Application Settings

Table 22: API Application Settings

Tab	Section	Field	Description
API	General	Allow Deprecated API Calls	If set to **False**, API applications will not be able to access earlier versions of API methods or other legacy API methods that have been replaced or updated. Many of the updated methods offer additional security measures, so this setting can reduce the risk of unauthorized API access, but may cause API applications written against these earlier versions to stop functioning correctly. If you do not have any such applications, this should be set to **False**. The default is True. For more information, see Versioning on page 852 in the *Keyfactor API Reference Guide*.

Tab	Section	Field	Description
API	General	API Throttling Interval (seconds)	The maximum rate at which API applications can make requests to the API. A larger value will mitigate risks from certain denial of service and brute-force/-dictionary attacks, but will limit the performance of applications needing to make multiple API calls. This can be set to zero to disable throttling.
API	Certificate Enrollment	Authorization Token Timeout	This is considered deprecated and may be removed in a future release.
API	Certificate Enrollment	Reverse Legacy Enrollment Chain Order	This is considered deprecated and may be removed in a future release.

Application Settings: SSH Tab

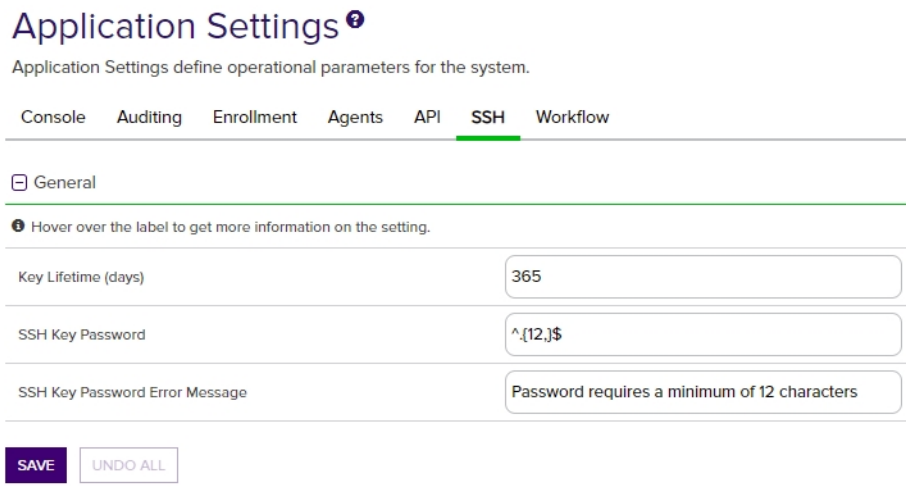


Figure 341: SSH Settings

Table 23: SSH Application Settings

Tab	Section	Field	Description
SSH	General	Key Lifetime (days)	The number of days for which an SSH key generated through My SSH Key (see Generating a New Key on page 536) or Service Account Keys (see Creating a Service Account Key on page 545) is considered valid. The default is 365 days.

Tab	Section	Field	Description
SSH	General	SSH Key Password	The regular expression against which the password entered when creating, rotating or downloading keys for both user SSH keys (My SSH Key on page 531) and service account SSH keys (Service Account Keys on page 542) will be validated. The default is a minimum of 12 characters configured as: `^.{12,}$`
SSH	General	SSH Key Password Error Message	The error message displayed to the user in the relevant SSH pages of the Keyfactor Command Management Portal when the password referenced does not match the regular expression defined for the password using the SSH Key Password setting.

Application Settings: Workflow Tab

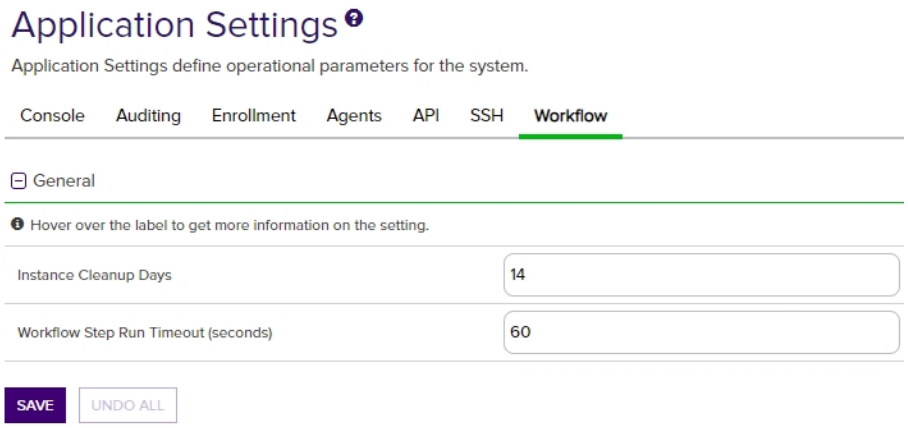


Figure 342: Workflow Settings

Table 24: Workflow Application Settings

Tab	Section	Field	Description
Workflow	General	Workflow Step Run Timeout (seconds)	The number of seconds a workflow instance step will be allowed to run before timing out and setting the instance to a status of Failed. The default is 60 seconds.
Workflow	General	Instance	The number of days to retain completed workflow

Tab	Section	Field	Description
		Cleanup Days	instances (successful or failed) before they are purged. The cleanup job runs daily at midnight. The default value is 14.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

2.1.11.2 Security Roles and Claims

There are several elements that make up Keyfactor Command Security infrastructure. To define your security design you will use these elements in combinations that meet your needs. You can limit user menu access through global permissions, and user certificate access through collection and certificate stores permissions.

- Security Roles—Management Portal and Keyfactor API Access Control

  Define the naming convention and structure of your security design by creating a name and description for your roles. These roles will then hold the definition of your security design based on the menu access, certificate collection access or certificate store access as applied to them. The roles will then be applied to users or groups to complete the security set-up. These roles are used to:

  - Grant access to the Management Portal and Keyfactor API, by selecting area access permissions for a role—for example, at what level of permission the user/group can access certificates functionality on the Keyfactor Command Management Portal. See Security Role Permissions on page 632 and Security Role Operations on page 683.
  - Grant certificate collection access by selecting role permissions per collection—at which level of permission the user/group can access collection functionality and/or which collections they can access. See Certificate Collection Permissions on page 627.
  - Grant certificate store container access by selecting role permissions per container—at which level of permission the user/groups can access certificate store functionality, and/or which stores they can access. See Container Permissions on page 629.

- Security Claims (formerly Identities)—Management Portal and Keyfactor API authentication.

  Assign combinations of **Roles** to users or groups to apply your security design to your users. See Security Claim Operations on page 692.

Security Roles

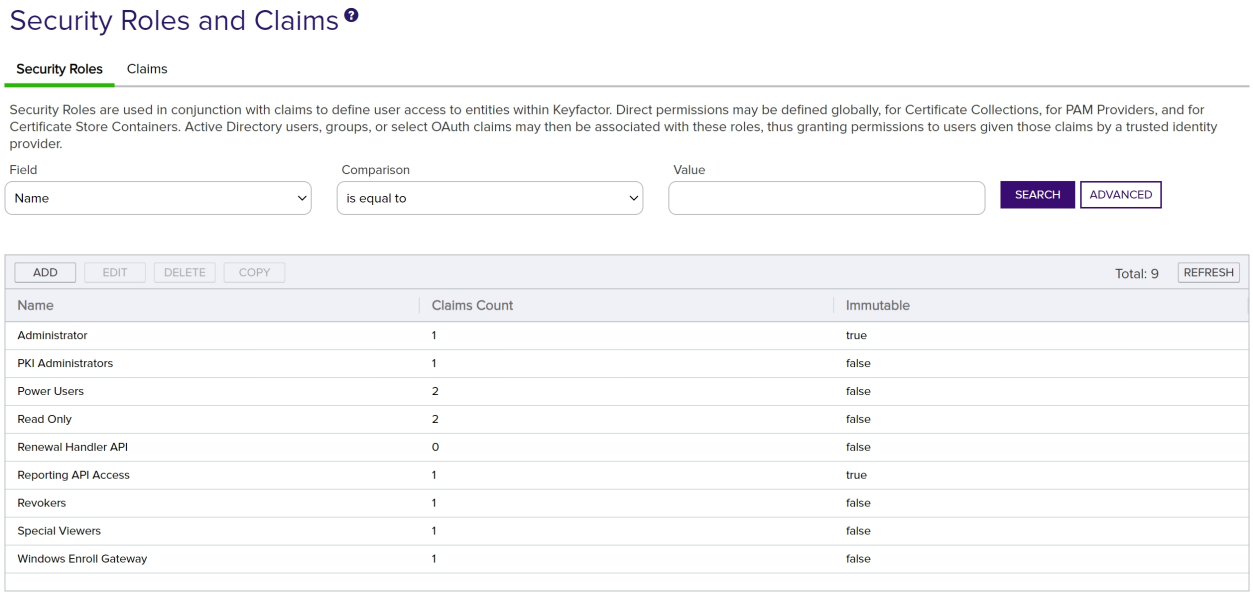


Figure 343: Security Roles

During the Keyfactor Command installation and configuration process, the security role **Administrator** is created (see Administrative Users Tab on page 2805). The **Administrator** role grants full permissions to the Management Portal and cannot be edited or deleted. If all users of the Management Portal should have full access to all features within the portal, this one role may be sufficient for your needs. However, if you would like to grant access to other users or limit the functionality available to those users, you need to add one or more new security roles for this purpose.

A **Reporting API Access** role is automatically created during installation to support the dashboard and reporting access required by the Logi Analytics Platform. The service account user associated with the IIS application pools on the Keyfactor Command Management Portal server (where Logi is installed) is automatically created as an identity and associated with this role.

Security Claims

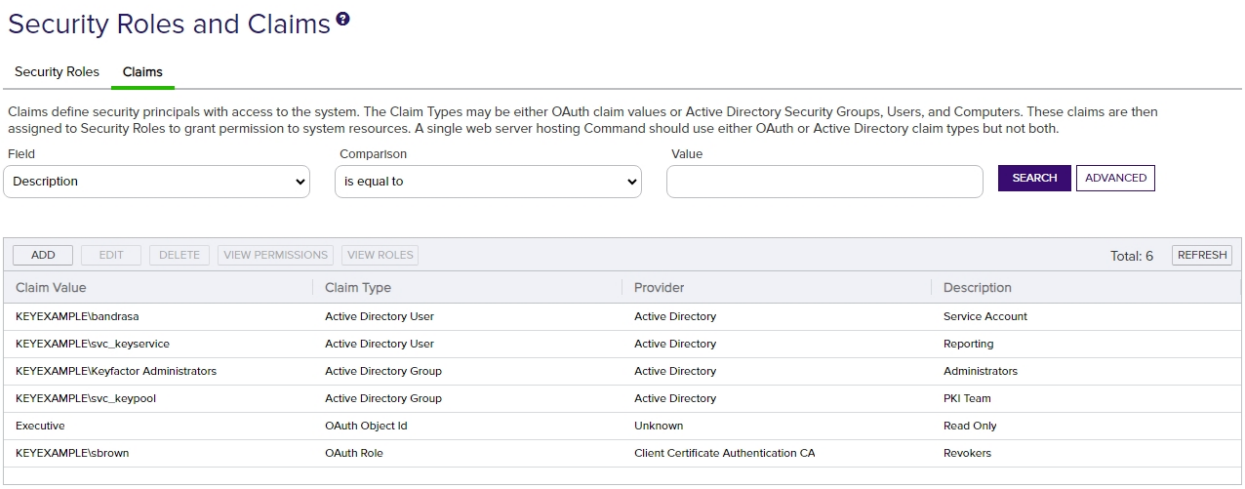


Figure 344: Security Claims

Claims are created in Keyfactor Command using users or groups. During the Keyfactor Command installation and configuration process, administrative security claims are created using the user and/or group records for your selected identity provider (either Active Directory or an alternative) on the Administrative Users tab of the configuration wizard (see Administrative Users Tab on page 2805). More than one user or group may be entered during configuration, if desired. Claims entered in the configuration wizard are associated with the **Administrators** role that grants all permissions to the Management Portal.

If you would like to grant access to other users but limit the functionality available to those users, you need to add one or more new security claims for this purpose and link them to one or more appropriate security roles. See Security Claim Operations on page 692.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Keyfactor Command Security Design Considerations

As you create your security design, be sure to cover the following:

- Determine the list of users, groups, and other entities who will have access to Keyfactor Command. Access in Keyfactor Command is based on identity provider users and groups or roles (see Identity Providers on page 754). These will be used to create **Security Claims** in Keyfactor Command to which **Security Roles** will be assigned.

Note: If you require only one layer of security (all users will have full access) you may wish to simply use the Administrator Role that was created during installation (see Administrative Users Tab on page 2805).

Note: When defining the users and groups you will use to form **claims**, consider whether you will have a one-to-one or one-to many relationship between **Claims** and **Roles**.

- Define the naming convention for **Security Roles**. Menu access and certificate security will be assigned to **Roles** which in turn will be applied to **Security Claims**.
- Determine the Keyfactor Command menu access and Keyfactor API access as well as the level of functionality you want to apply to each **Role** using the permissions information found Security Role Permissions on page 632.
- If you need a further level of control beyond Security Roles, consider **Permission Sets**. Permission sets are containers which allow you to organize roles and compartmentalize permissions. They can only be configured through the Keyfactor API (see Permission Sets on page 1959).
- Determine certificate security based on collections and certificate store permissions based on containers, if any. See below for more information.

Certificate Store Container Permissions

When designing a container permission scheme, you need to think first about whether you want your users to have access to all the certificate stores in your Keyfactor Command database or whether you need to limit your users to having access to only a subset of your stores. If you're comfortable granting access to all the stores, you can use the global Read permission. If you're not comfortable with this, you need to use container-level permissions and grant Read permissions on a container-by-container basis. These can be granted separately on a group-by-group (or user-by-user) basis, so group A can be granted global Read while group B is only granted Read to a certain container.

Next, you need to think about what you want your users to be able to do with the stores they have access to. By granting Read access to the stores, you're allowing your users to browse to the certificate stores page and see all the stores and containers that they've been granted access to, but they can perform no operations related to the stores. These are controlled with additional permissions (see below) that can also be set either globally or on a container-by-container basis. You can combine global and container-level security.

Example: You've decided that you need to use container-level security at the *Read* level on three different containers rather than granting global *Read* to your Web Server Managers group. You want these users to be able to push new certificates out to certificate stores in the IIS Personal, PEM and Java containers but not to stores on your F5 and NetScaler devices. You could either grant them the *Schedule* permission on a container-by-container basis or you could grant them the global *Schedule* permission for Certificate Store Management. Since the users have neither the global *Read* permission nor container permission for the containers for the F5 and NetScaler devices, these two settings would accomplish the same goal.

In addition to the permissions that must be considered when designing a permission scheme for certificate stores, you must also give consideration to permissions for certificates. Users must have permissions to certificates in order to use the certificate store operations. See Certificate Collection Permissions on the next page and Container Permissions on page 629.

Note: Setting permissions on a container-by-container basis automatically grants the lower permissions (e.g. setting *Schedule* automatically grants *Read*). The same is not true for permissions set at the global level.
Any containers that do not have container-by-container permissions applied fall back to the global permissions, if any global permissions have been set.

Certificate and Collection-by-Collection Permissions

When designing a certificate permission scheme, you need to think first about whether you want your users to have access to all the certificates in your Keyfactor Command database or whether you need to limit your users to having access to only a subset of your certificates. If you're comfortable granting access to all the certificates, you can use the global Read permission. If you're not comfortable with this, you need to use collection-level permissions and grant Read permissions on a collection-by-collection basis. These can be granted separately on a group-by-group (or user-by-user) basis, so group A can be granted global Read while group B is only granted Read to a certain collection.

Next, you need to think about what you want your users to be able to do with the certificates they can view. There are certificate operation permissions (see Certificate Collection Permissions on the next page) that you can set that control what your users can do with the certificates. These can be set either globally or on a collection-by-collection basis. You can combine global and collection-level security.

Example: You've decided that you need to use collection-level security at the Read level on four different collections to grant Read access to your PKI Help Desk group and will not grant them global Read permissions. You also want these users to be able to edit the metadata fields of the certificates in all four of these collections. You could either grant them the Edit Metadata permission on a collection-by-collection basis or you could grant them the global Edit Metadata permission. Since the users don't have the global Read permission (and thus can't read the other collections), these two settings would accomplish the same goal.

At the global level, the **Certificates** Read role permission grants access to both the certificate search page and all certificate collections. Users who have been granted only collection-level Read permissions and not global Read permissions have access only to the collections to which they have been granted access and not to the certificate search page. See Security Role Permissions on page 632 and Certificate Collection Permissions on the next page.

In addition to the **Certificates** role permissions that must be considered when designing a permission scheme for certificates, you must also give consideration to the **Certificate Collections** and **Certificate Store Management** global role permissions.

- Enabling the Certificate Collections Modify global role permission allows users to use the Save, Save As and Delete buttons for a collection. This allows users to create new certificate collections based on existing collections (Save As), delete existing collections (Delete), or modify select settings about an existing collection (Save). Typically, Certificate Collections permissions would only be granted to users who also had at least global Read permissions to allow them to do certificate searches from which to create new collections.
- You will need to consider the Certificate Store Management role permissions if you use certificate stores and want any of your limited access users to make use of the Add to Certificate Store, Remove from Certificate Store, or Renew/Reissue operations on certificates. These certificate operations are only available to users who have also been granted the Read and Schedule role permissions for Certificate Store Management. Permissions to certificate stores can be granted either globally or via container security (see Certificate Collection Permissions below and Container Permissions on page 629).

Certificate Collection Permissions

Permissions on certificates and their collections can be controlled at two levels—system-wide at the certificates-collections level and on a collection-by-collection basis. When designing a certificate permission scheme, you may use entirely system-wide permissions or you may use a combination of system-wide permissions and collection permissions. Both system-wide and collection permissions are configured through Security Roles (see Security Role Operations on page 683).

System-wide certificate permissions are controlled using the **Certificates** / **Collections** role permission.

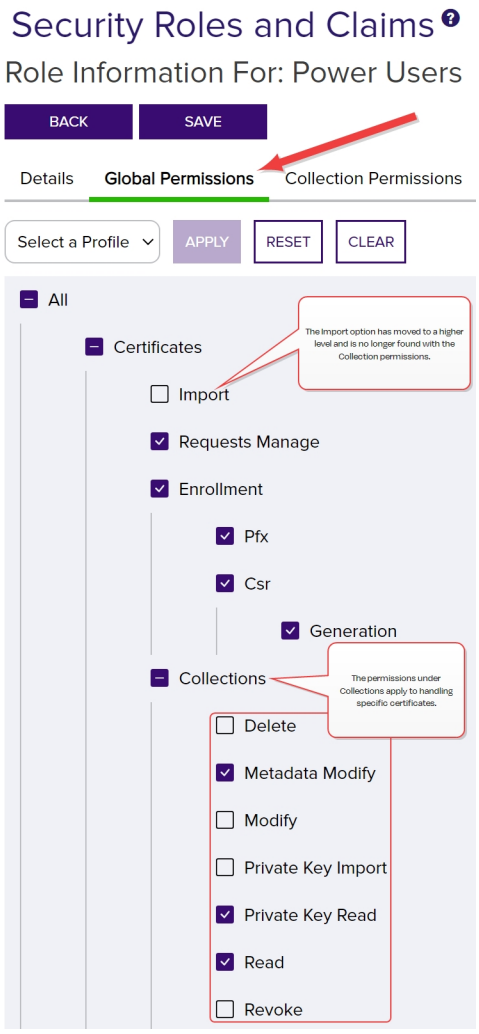


Figure 345: Certificate Collection System-wide Permissions

Note: The *Import* permission that was previously among the permissions controlled at the system-wide certificate collection level, has moved to a higher level in the permission tree and is no longer found at the collection level.

Permissions for managing collections (modification of existing collections or creation of new collections) are controlled with the system-wide **Certificates** / **Collections** role *Modify* permission. The Certificates Collections Modify permission allows a user to edit the configuration settings that make up the collections (e.g. the query that defines the collection). The permissions set on the Collection Permissions tab allow a user to act on the certificates in the collection.

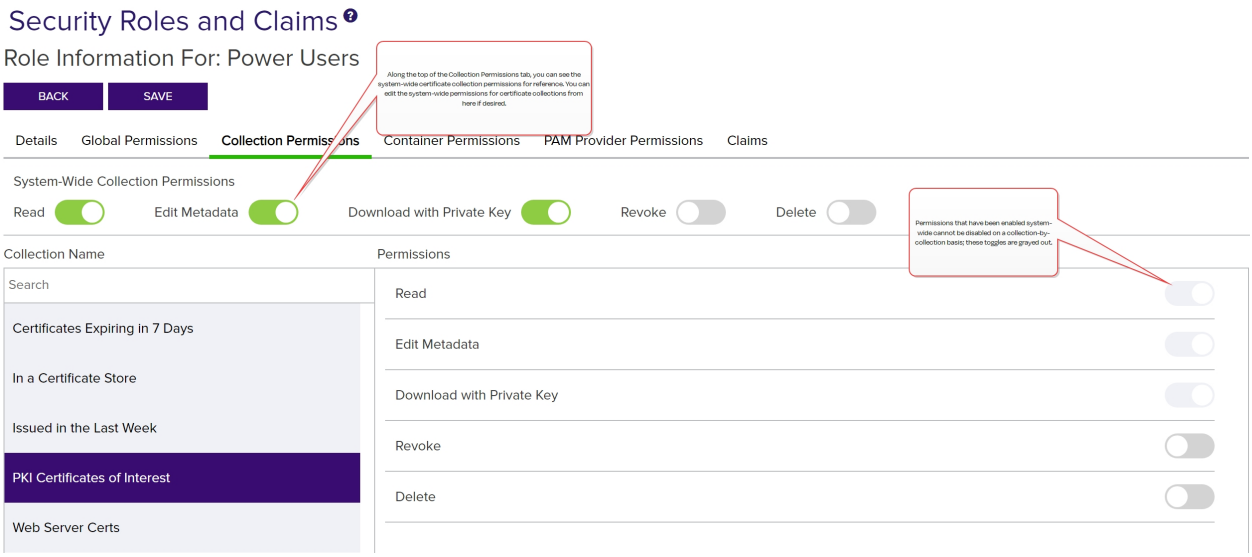


Figure 346: Certificate Collection per Collection Permissions

Certificate-related permissions can be granted globally or on a collection basis. Both options share the same permission options (see Certificates on page 646) except system-wide certificate permissions have the additional role permissions of *Modify* and *Private Key Import*, which cannot be assigned at the collection level.

Any certificate collections that do not have collection-level permissions applied fall back to the system-wide permissions, if any system-wide permissions have been set for that role.

For more information about configuring collection-level permissions, see Collection Permissions Tab on page 685.

Container Permissions

Permissions on certificate stores are controlled at two levels—system-wide and on a certificate store container-by-container basis. When designing a certificate store permission scheme, you may use entirely system-wide permissions or you may use a combination of system-wide permissions and container permissions. Both system-wide and container permissions are configured through Security Roles (see Security Role Operations on page 683).

System-wide certificate store permissions are controlled with the **Certificate Store Management** role permissions on the **Global Permissions** tab of the Security Role Information dialog.

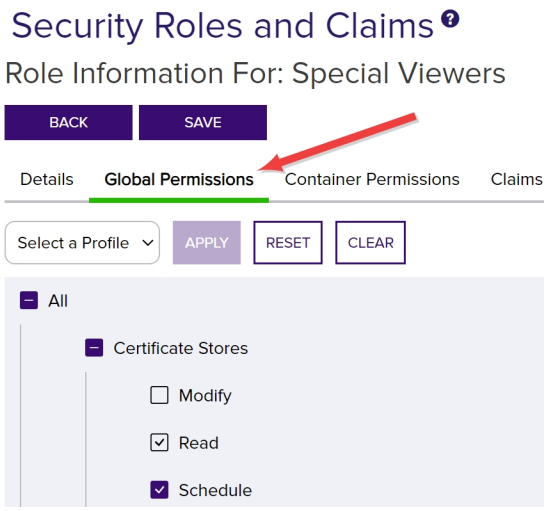


Figure 347: Certificate Store Management—Global Permissions

Container-by-container permissions are set on the **Container Permissions** tab of the Role Information dialog for each container by name using the same set of permissions.

Any containers that do not have container-by-container permissions applied fall back to the system-wide permissions, if any system-wide permissions have been set for that role.

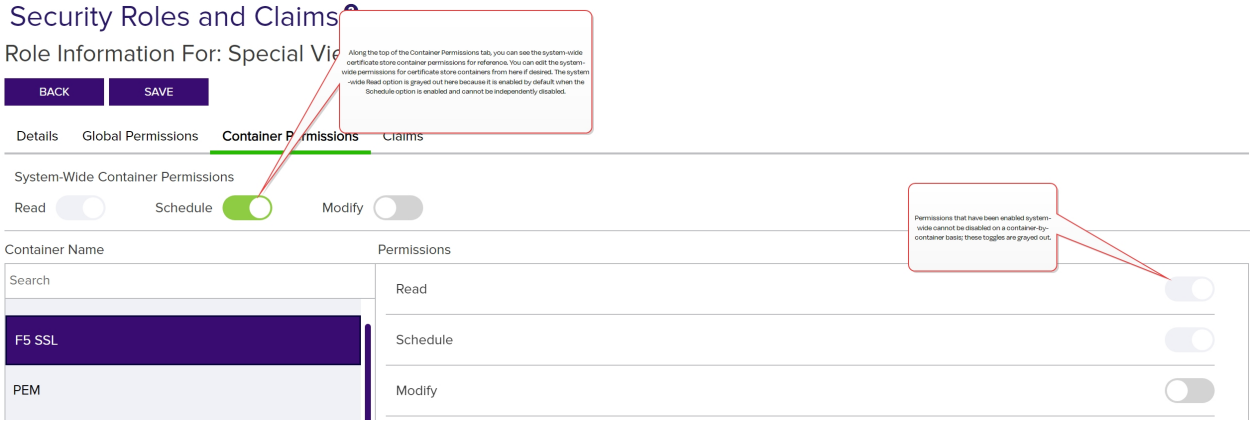


Figure 348: Certificate Store Management - Container Permissions

Certificate store permissions can be granted system-wide or on a container basis. Both options share the same permission options (see Certificate Stores on page 637). Container permissions work in conjunction with many other security permissions to control access to certificate store related functionality.

For more information about configuring container-level permissions, see Container Permissions Tab on page 686.

Tip: See the detailed tip sections of Certificate Operations on page 45, Certificate Store Operations on page 413 and Certificate Store Types on page 700 for more information regarding which combination of security permissions are required for various operations.

PAM Permissions

Permissions on PAM can be controlled at two levels—system-wide and on a provider-by-provider basis. When designing a PAM permission scheme, you may use entirely system-wide permissions or you may use a combination of system-wide permissions and provider-level permissions. Both system-wide and provider-level permissions are configured through Security Roles (see Security Role Operations on page 683).

Tip: Users also need *Read* permissions for **System Settings** to be able to access the *System Settings Icon* > *Privileged Access Management* page, and *Modify* permissions to modify the settings.

System-wide PAM permissions are controlled using the **Privileged Access Management** role permission. These permissions control which users have access to viewing and managing any PAM providers you will use in your Keyfactor Command implementation.

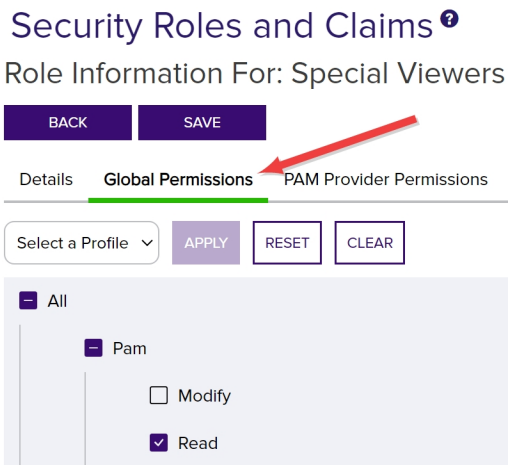


Figure 349: Global PAM Permissions

PAM provider-level permissions are controlled with the optional provider-by-provider permissions on the **PAM Provider Permissions** tab of the Security Role Information dialog. The permissions set on the PAM Provider Permissions tab allow a user to access only the referenced PAM provider when selected.

Any PAM providers that do not have provider-level permissions applied fall back to the system-wide permissions, if any system-wide permissions have been set for that security role.

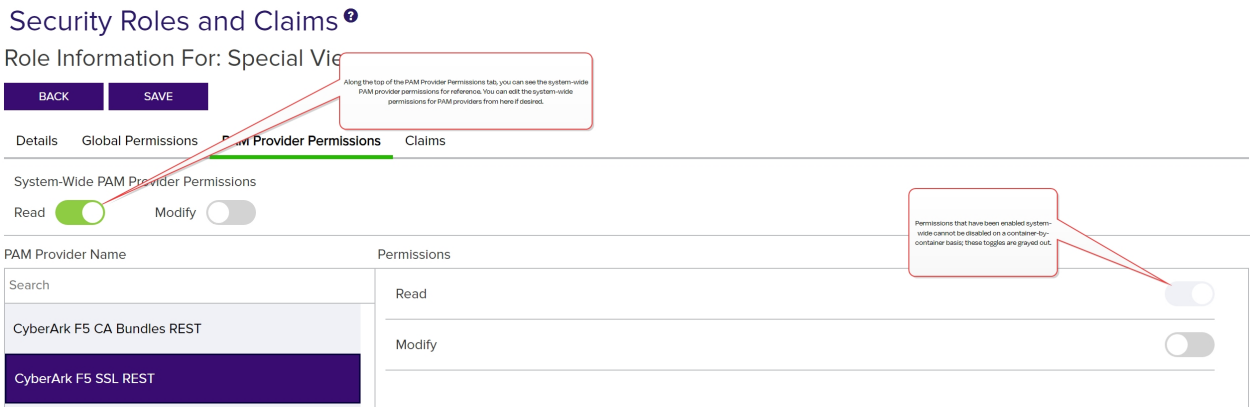


Figure 350: PAM Provider Permissions

PAM permissions can be granted system-wide or on a provider basis. Both options share the same permission options (see Privileged Access Management (PAM) on page 663).

Security Role Permissions

The Security Role Permissions that are available to be assigned to security roles within Keyfactor Command are documented below. For release 11.0 of Keyfactor Command, a new permission structure has been introduced. Users of Keyfactor Command through the Management Portal will not see much difference between the older model and the newer model, as the changes are largely behind the scenes. Users of Keyfactor Command through the Keyfactor API will need to understand the new model. Some Keyfactor API endpoints (e.g. v1 Security Roles endpoints) still use the older permission model. Other Keyfactor API endpoints (e.g. v2 Security Roles endpoints) use the newer permission model.

Version Two Permission Model

The version two permission model was introduced in Keyfactor Command 11.0 and is used when setting security permissions in the Management Portal, with v2 Security Roles Keyfactor API endpoints, and with Keyfactor API Permission Set endpoints.

In the new model, permissions are built from access control strings, which are structured to support permission inheritance. Generally speaking, the more you add to an access control string, the less privilege you are granting to a user in that area of the product. For example, the following access control string grants full control to the entire product:

```
/
```

Add a certificates level to this, and now you've limited this to full control of just functions related to certificates in the product (which would include enrollment, for example):

```
/certificates/
```

Add a collections level to this, and now you've limited this further to full control of just options that can be found on the Certificates menu item in the Management Portal, including certificates both in collections and found by direct search, certificate import, and certificate collection management:

```
/certificates/collections/
```

Add a read to this, and now you've limited this to just read for items on the Certificates menu:

```
/certificates/collections/read/
```

Add a certificate collection ID to this, and now you've locked this down to just read on just the certificates in the certificate collection with ID 5:

```
/certificates/collections/read/5/
```

When you apply permissions through the Management Portal, these access control strings are applied for you based on the selections you make in the Role Information dialog when assigning permissions to a role (see Security Role Operations on page 683). When you apply permissions through the Keyfactor API using a newer endpoint (e.g. v2 Security Roles endpoints), you need to specify these access control strings.

Access control strings that are shown below with a # refer to a specific granular ID to which permissions should be granted. When used, they must be specified with an integer in place of the #. For example, use:

```
/certificate_stores/read/4/
```

To refer to the certificate store container with ID 4, not:

```
/certificate_stores/read/#/
```

Note: Access control strings always begin and end with a /. If you specify an access control string in the Keyfactor API without the leading or trailing slash, it will not be recognized.

Agents

Table 25: Agents Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Agents	/agents/	Users can view and modify agent auto-registration settings, Mac auto-enroll management settings, and

Permission Tab	Portal Permission	API Permission	Description
			orchestrator management and jobs.
Global	Agents > Auto-Registration	/agents/auto_registration/	Users can view and modify the agent auto-registration settings.
Global	Agents > Auto-Registration > Modify	/agents/auto_registration/modify/	Users can modify the agent auto-registration settings.
Global	Agents > Auto-Registration > Read	/agents/auto_registration/read/	Users can view the agent auto-registration settings; Users must also have Read permissions for Agent Management.
Global	Agents > Management	/agents/management/	Users can view and modify orchestrator management and jobs.
Global	Agents > Management > Modify	/agents/management/modify/	Users can access the Management Portal areas and API endpoints to: • Manage orchestrators, including approving and disapproving them • Unschedule and reschedule orchestrator jobs
Global	Agents > Management > Read	/agents/management/read/	Users can access the Management Portal areas and API endpoints to: • View orchestrators, including filtering the orchestrator management grid

Permission Tab	Portal Permission	API Permission	Description
			• View orchestrator jobs, including status, schedules, failures and warnings
Global	Agents > Management > Mac	/agents/management/mac/	Users can view and modify Mac auto-enroll management settings.
Global	Agents > Management > Mac > Auto-enrollment	/agents/management/mac/auto-enrollment/	Users can view and modify Mac auto-enroll management settings.
Global	Agents > Management > Mac > Auto-enrollment > Management	/agents/management/mac/auto-enrollment/management/	Users can view and modify Mac auto-enroll management settings.
Global	Agents > Management > Mac > Auto-enrollment > Management > Modify	/agents/management/mac/auto-enrollment/management/modify/	Users can modify the Mac auto-enroll management settings.
Global	Agents > Management > Mac > Auto-enrollment > Management > Read	/agents/management/mac/auto-enrollment/management/read/	Users can view the Mac auto-enroll management settings.

Application Settings

Table 26: Application Settings Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Application Settings	/application_settings/	Users can view and modify the application settings.
Global	Application Settings > Modify	/application_settings/modify/	Users can modify the application settings.
Global	Application Settings > Read	/application_settings/read/	Users can view the application settings.

Auditing

Table 27: Auditing Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Auditing	/auditing/	Users can access the Audit Log page in the Management Portal, and will be able to make API requests to obtain data from the audit log (query, etc.).
Global	Auditing > Read	/auditing/read/	Users can access the Audit Log page in the Management Portal, and will be able to make API requests to obtain data from the audit log (query, etc.).

Certificate Authorities

Note: This permission was previously bundled together with certificate template permissions and known as PKI Management.

Table 28: Certificate Authorities Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Certificate Authorities	/certificate_authorities/	Users can view and modify certificate authority records. Users can view, test, and modify revocation monitoring

Permission Tab	Portal Permission	API Permission	Description
			settings.
Global	Certificate Authorities > Modify	/certificate_authorities/modify/	Users can modify certificate authority and revocation monitoring settings to: • Import, add, edit, and delete certificate authorities. • Configure CA health monitor and threshold alert recipients. • Import CAs. • Add, edit, delete, and test revocation monitoring endpoints. For access in the Management Portal, this also requires the *Monitoring > Alerts > Read* permission. • Configure revocation monitoring schedules and recipients. For access in the Management Portal, this also requires the *Monitoring > Alerts > Read* permission.
Global	Certificate Authorities > Read	/certificate_authorities/read/	Users can view certificate authority records. Users can view revocation monitoring settings, CA health monitoring and threshold alert recipients and schedules.

Certificate Stores

Table 29: Certificate Stores Security Role Permissions v2

See Container Permissions on page 629, Certificate Operations on page 45, Certificate Store Types on page 700 and Certificate Store Operations on page 413 for more information.

Permission Tab	Portal Permission	API Permission	Description
Global	Certificate Stores	/certificate_stores/	Users can view and manage all certificate stores and add certificates to certificate stores, renew/reissue certificates, and remove certificates from certificate stores for all certificate stores.

Permission Tab	Portal Permission	API Permission	Description
Global	Certificate Stores > Modify	/certificate_stores/-modify/	Users with the *Modify* role permission for either **Certificate Stores** or a container (#) can view the certificate stores grid and the containers grid and use the following operations on these pages. The *Modify* permission must be granted in conjunction with either **Certificate Stores** *Read* or container (3) *Read* for full functionality. Users must have global **Certificate Stores** *Read* and *Modify* permissions to access the discover tab and use the functions on it. Users with *Modify* permissions granted at the container level can perform these *certificate store* operations (in addition to those available with *Read* permissions): • **Add**—When a new certificate store is added, only containers on which the user has *Modify* permissions will appear in the container dropdown and the newly created certificate store must be added to one of these if the user does not have global certificate store management permissions. • **Edit**—Only certificate stores in containers on which the user has *Modify* permissions will be editable. • **Delete**—Only certificate stores in containers on which the user has *Modify* permissions will be available for deletion. • **Assign Container**—The container assigned to a certificate store can only be changed if the store is currently in a container on which the user has *Modify* permissions, and the store can only be assigned

Permission Tab	Portal Permission	API Permission	Description
			to another container on which the user has *Modify* permissions. The user must have *Modify* permissions on both the source container and the target container when moving a certificate store from one container to another. • **Set New Password**—Only certificate stores in containers on which the user has *Modify* permissions will be available for password reset. • **Reenrollment**—Only certificate stores in containers on which the user has *Schedule* permissions, along with *Enroll CSR* permissions (see Certificate Enrollment on page 672) will be available for reenrollment. Note that this permission does not control additions of certificates to certificate stores.
Container	Certificate Stores > Modify	/certificate_stores/-modify/#/	Users with the *Modify* role permission for either **Certificate Stores** or a container (#) can view the certificate stores grid and the containers grid and use the following operations on these pages. The *Modify* permission must be granted in conjunction with either **Certificate Stores** *Read* or container (3) *Read* for full functionality. Users must have global **Certificate Stores** *Read* and *Modify* permissions to access the discover tab and use the functions on it. Users with *Modify* permissions granted at the container level can perform these *certificate store* operations (in addition to those available with *Read* permissions):

Permission Tab	Portal Permission	API Permission	Description
			• **Add**—When a new certificate store is added, only containers on which the user has *Modify* permissions will appear in the container dropdown and the newly created certificate store must be added to one of these if the user does not have global certificate store management permissions. • **Edit**—Only certificate stores in containers on which the user has *Modify* permissions will be editable. • **Delete**—Only certificate stores in containers on which the user has *Modify* permissions will be available for deletion. • **Assign Container**—The container assigned to a certificate store can only be changed if the store is currently in a container on which the user has *Modify* permissions, and the store can only be assigned to another container on which the user has *Modify* permissions. The user must have *Modify* permissions on both the source container and the target container when moving a certificate store from one container to another. • **Set New Password**—Only certificate stores in containers on which the user has *Modify* permissions will be available for password reset. • **Reenrollment**—Only certificate stores in containers on which the user has *Schedule* permissions, along with *Enroll CSR* permissions (see Certificate Enrollment on page 672) will be available for reenrollment.

Permission Tab	Portal Permission	API Permission	Description
			Note that this permission does not control additions of certificates to certificate stores.
Global	Certificate Stores > Read	/certificate_stores/read/	Users with the *Read* global role permission for either **Certificate Store** or a specific container (#) can view the certificate stores grid and the containers grid and see all the certificate stores and store types. They can perform no operations on the certificate stores or containers from the certificate stores page. If the users also have global *Read* permissions for **Certificates** > **Collections**, they can view inventory for a certificate store by highlighting a certificate store in the certificate store grid and clicking the **View Inventory** button. Users with *Read* permissions granted at the container level can perform these *certificate store* operations: • **View**—Only certificate stores in containers on which the user has *Read* permissions will appear in the certificate stores grid. Users can open each certificate store to view the details of it but cannot change them. • **View Inventory**—Only certificate stores in containers on which the user has *Read* permissions will be available with the View Inventory option. Users with *Read* permissions granted at the container level can perform these *container* operations: • **View**—Only containers on which the user has *Read* permissions will

Permission Tab	Portal Permission	API Permission	Description
			appear in the containers grid. Users can open each container to view the details of it but cannot change them.
Container	Certificate Stores > Read	/certificate_stores/read/#/	Users with the *Read* global role permission for either **Certificate Stores** or a specific container (#) can view the certificate stores grid and the containers grid and see all the certificate stores and store types. They can perform no operations on the certificate stores or containers from the certificate stores page. If the users also have global *Read* permissions for **Certificates > Collections**, they can view inventory for a certificate store by highlighting a certificate store in the certificate store grid and clicking the **View Inventory** button. Users with *Read* permissions granted at the container level can perform these *certificate store* operations: • **View**—Only certificate stores in containers on which the user has *Read* permissions will appear in the certificate stores grid. Users can open each certificate store to view the details of it but cannot change them. • **View Inventory**—Only certificate stores in containers on which the user has *Read* permissions will be available with the View Inventory option. Users with *Read* permissions granted at the container level can perform these *container* operations: • **View**—Only containers on which the

Permission Tab	Portal Permission	API Permission	Description
			user has *Read* permissions will appear in the containers grid. Users can open each container to view the details of it but cannot change them.
Global	Certificate Stores > Schedule	/certificate_stores/schedule/	Users with the *Schedule* and *Read* role permission for either **Certificate Stores** or a container (#) can use the **Add to Certificate Store**, **Remove from Certificate Store** from the certificate search page, and **Schedule** from the certificate stores page. Users must have either global *Read* for **Certificates** > **Collections** or have been granted access to one or more certificate collections via collection-level permissions (see Certificate Collection Permissions on page 627) to use the add, remove, and renew/re-issue certificate options. To renew/re-issue certificates, users must also have been assigned to a role with **Certificates** > **Enrollment** permissions (*PFX* and/or *CSR* (see Certificate Enrollment on page 672)) and **Agents** > **Management**-*Read* Users with *Schedule* and *Read* permission may perform this operation on the *certificate store* or *container* grid. • **Schedule Inventory**—Only certificate stores in containers on which the user has *Schedule*permissions will be available for scheduling inventory. Users with *Schedule* and *Read* permissions granted at the container level can perform these *certificate* operations: • **Add to Certificate Store**—Only

Permission Tab	Portal Permission	API Permission	Description
			certificate stores in containers on which the user has *Schedule* permissions will be available for adding a certificate to in certificate collections or certificate search. • **Remove from Certificate Store**—Only certificate stores in containers on which the user has *Schedule* permissions will be available for removing a certificate from in certificate collections or certificate search. • **Renew/Reissue**—Only certificate stores in containers on which the user has *Schedule* permissions will be available for renewing or reissuing a certificate in certificate collections or certificate search. .
Container	Certificate Stores > Schedule	/certificate_stores/schedule/#/	Users with the *Schedule* and *Read* role permission for either **Certificate Stores** or a container (#) can use the **Add to Certificate Store**, **Remove from Certificate Store** from the certificate search page, and **Schedule** from the certificate stores page. Users must have either global *Read* for **Certificates** > **Collections** or have been granted access to one or more certificate collections via collection-level permissions (see Certificate Collection Permissions on page 627) to use the add, remove, and renew/reissue certificate options. To renew/reissue certificates, users must also have been assigned to a role with **Certificates** > **Enrollment** permissions (*PFX* and/or *CSR* (see Certificate Enrollment on page 672)) and **Agents** >

Permission Tab	Portal Permission	API Permission	Description
			Management-*Read* Users with *Schedule* and *Read* permission may perform this operation on the *certificate store* or *container* grid. • **Schedule Inventory**—Only certificate stores in containers on which the user has *Schedule*permissions will be available for scheduling inventory. Users with *Schedule* and *Read* permissions granted at the container level can perform these *certificate* operations: • **Add to Certificate Store**—Only certificate stores in containers on which the user has *Schedule* permissions will be available for adding a certificate to in certificate collections or certificate search. • **Remove from Certificate Store**—Only certificate stores in containers on which the user has *Schedule* permissions will be available for removing a certificate from in certificate collections or certificate search. • **Renew/Reissue**—Only certificate stores in containers on which the user has *Schedule* permissions will be available for renewing or reissuing a certificate in certificate collections or certificate search.

Certificate Templates

Note: This permission was previously bundled together with certificate authority permissions and known as PKI Management.

Table 30: Certificate Templates Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Certificate Templates	/certificate_templates/	Users can view and modify certificate template records.
Global	Certificate Templates > Read	/certificate_templates/read/	Users can view certificate template records.
Global	Certificate Templates > Modify	/certificate_templates/modify/	Users can modify certificate template settings to import, edit, and configure system settings for certificate templates.

Certificates

See Certificate Collection Permissions on page 627 for additional information.

Table 31: Certificates Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Certificates	/certificates/	Users can view, modify, and act upon everything certificate-related, including certificates in collections, certificates found in a search that are not in a collection, certificate import, certificate enrollment, and pending certificate request management.
Global	Certificates > Import	/certificates/import/	Users can import certificates using the Management Portal Add Certificate page or the Keyfactor API POST /Certificates/Import

Permission Tab	Portal Permission	API Permission	Description
			method. Users who also have Read permissions for Certificate Store Management or container permissions can add certificates to certificate stores from Add Certificate. **Note:** This permission was controlled at the global certificate collection level in previous versions of Keyfactor Command, but has moved to a higher level separate from collections.
Global	Certificates > Requests Manage	/certificates/requests/manage/	Users can use the Pending CSRs page in the Management Portal and the equivalent API functions.
Global	Certificates > Enrollment	/certificates/enrollment/	Users can use all the enrollment-related functions, including CSR generation, CSR enrollment, and PFX enrollment.
Global	Certificates >	/certificates/enrollment/pfx/	Users can use the

Permission Tab	Portal Permission	API Permission	Description
	Enrollment > Pfx		PFX Enrollment page in the Management Portal and the equivalent API functions.
Global	Certificates > Enrollment > Csr	/certificates/enrollment/csr/	Users can use the CSR Enrollment page in the Management Portal and the equivalent API functions.
Global	Certificates > Enrollment > Csr > Generation	/certificates/enrollment/csr/generation/	Users can use the CSR Generation page in the Management Portal and the equivalent API functions.
Global	Certificates > Collections	/certificates/collections/	Users can view, modify, and act upon certificate-related functions including certificates in collections and certificates found in a search that are not in a collection.
Global	Certificates > Collections > Delete	/certificates/collections/delete/	Users can delete certificates and, if applicable, the private keys of the certificates from the Keyfactor Command database for any certificates.
Collection	Certificates > Collections > Delete	/certificates/collections/delete/#/	Users can delete certificates and, if applicable, the private keys of the certificates from the

Permission Tab	Portal Permission	API Permission	Description
			Keyfactor Command database for certificates in the specified certificate collection.
Global	Certificates > Collections > Metadata Modify	/certificates/collections/metadata/modify/	Users can modify certificate metadata for certificates in the Certificate Details on page 19 dialog (only information on the metadata tab can be edited) and the equivalent API functions for any certificates.
Collection	Certificates > Collections > Edit Metadata	/certificates/collections/metadata/modify/#/	Users can modify certificate metadata for certificates in the Certificate Details on page 19 dialog (only information on the metadata tab can be edited) and the equivalent API functions for certificates in the specified certificate collection.
Global	Certificates > Collections > Modify	/certificates/collections/modify/	Users can add or edit certificate collections. See Certificate Collection Permissions on page 627 for more information. **Note:** This permission cannot be applied at the

Permission Tab	Portal Permission	API Permission	Description
			certificate collection level.
Global	Certificates > Collections > Private Key Import	/certificates/collections/private_key/import/	Users can save the private key for the certificate in the Keyfactor Command database. Users with this role can add a certificate with an associated private key through the Add Certificate option under the Certificate Locations menu (see Add Certificate on page 74) and the private key will be stored in the Keyfactor Command database. Users must also be granted the *Import* role in order to be able to use the Add Certificate feature. **Note:** This permission cannot be applied at the certificate collection level.
Global	Certificates > Collections > Download with Private Key	/certificates/collections/private_key/read/	Users can download the certificates with their private key for all certificates.

Permission Tab	Portal Permission	API Permission	Description
Collection	Certificates > Collections > Private Key Read	/certificates/collections/private_key/read/#/	Users can download the certificates with their private key for certificates in the specified certificate collection.
Global	Certificates > Collections > Read	/certificates/collections/read/	Users can view any certificates, including certificate history, and can download certificates. Users who also have Read permissions for Certificate Store Management or certificate store container permissions can add certificates to certificate stores from Certificate Search and Certificate Collections. The certificate operations possibly available to users with this permission are: • Add to Certificate Store (Also requires the *Certificate Stores > Schedule* permission at either the global or container level) • Display • Download • Get CSV • Identity Audit (Also requires the

Permission Tab	Portal Permission	API Permission	Description
			Security > Read permission) • Include Revoked checkbox • Include Expired checkbox • Renew (Also requires the *Certificates > Enrollment > Pfx* permission and *Certificate Stores > Schedule* permission at either the global or container level) • Remove from Certificate Store (Also requires the *Certificate Stores > Schedule* permission at either the global or container level) Users with **global Read** role permissions can browse to Certificate Search in the Management Portal and view all saved certificate collections. They can view any certificate in the Keyfactor Command database and are not limited to just those returned by select collections. Users with this permission can view

Permission Tab	Portal Permission	API Permission	Description
			the certificates returned by searches and open the details of the certificates.
Collection	Certificates > Collections > Read	/certificates/collections/read/#/	Users can view certificates in the specified certificate collection, including certificate history, and can download certificates. Users who also have Read permissions for Certificate Store Management or certificate store container permissions can add the certificates in the collection to certificate stores from Certificate Search and Certificate Collections. The certificate operations possibly available to users with this permission are: • Add to Certificate Store (Also requires the *Certificate Stores > Schedule* permission at either the global or container level) • Display • Download • Get CSV • Identity Audit

Permission Tab	Portal Permission	API Permission	Description
			(Also requires the *Security > Read* permission) • Include Revoked checkbox • Include Expired checkbox • Renew (Also requires the *Certificates > Enrollment > Pfx* permission and *Certificate Stores > Schedule* permission at either the global or container level) • Remove from Certificate Store (Also requires the *Certificate Stores > Schedule* permission at either the global or container level) Users with **collection-level Read** role permissions on a collection will see the collections to which they have been granted access appear on the Certificate Collections menu (if they have been configured to appear on the menu—see Certificate Collection Manager on page 85). The

Permission Tab	Portal Permission	API Permission	Description
			users will be able to view all the certificates in the collections and open the details of the certificates.
Global	Certificates > Collections > Revoke	/certificates/collections/revoke/	Users can revoke any certificates through Keyfactor Command. This includes certificates that have been issued by a Microsoft or EJBCA CA configured for synchronization or by a cloud-based certificate vendor that is managed via a Keyfactor certificate gateway. ⚠ **Important:** In order to successfully revoke certificates, the service account under which the Keyfactor Command application pool is running must be granted “Issue and

Permission Tab	Portal Permission	API Permission	Description
			⚠ Manage Certificates" and "Manage CA" permissions to the CA database as per Create Groups to Control Access to Keyfactor Command Features on page 2762, or, if delegation is configured for the CA, the user executing the revoke must have the "Issue and Manage Certificates" permissions while the application pool service account has the "Manage CA" permissions. If you are using explicit credentials to authenticate your CA (see Adding or Modifying a

Permission Tab	Portal Permission	API Permission	Description
			⚠ CA Record on page 354), it is the user specified on the CA configuration in Keyfactor Command who must have permissions on the CA.
Collection	Certificates > Collections > Revoke	/certificates/collections/revoke/#/	Users can revoke certificates in the specified certificate collection through Keyfactor Command. This includes certificates that have been issued by a Microsoft or EJBCA CA configured for synchronization or by a cloud-based certificate vendor that is managed via a Keyfactor certificate gateway. ⚠ **Important:** In order to successfully revoke certificates, the

Permission Tab	Portal Permission	API Permission	Description
			⚠ service account under which the Keyfactor Command application pool is running must be granted “Issue and Manage Certificates” and “Manage CA” permissions to the CA database as per Create Groups to Control Access to Keyfactor Command Features on page 2762, or, if delegation is configured for the CA, the user executing the revoke must have the “Issue and Manage Certificates” permissions while the application pool service

Permission Tab	Portal Permission	API Permission	Description
			⚠ account has the “Manage CA” permissions. If you are using explicit credentials to authenticate your CA (see Adding or Modifying a CA Record on page 354), it is the user specified on the CA configuration in Keyfactor Command who must have permissions on the CA.

Dashboard

Table 32: Dashboard Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Dashboard	/dashboard/	Users can view the panels, including the risk header, on their personalized dashboard and add and remove the customizable panels.
Global	Dashboard > Read	/dashboard/read/	Users can view the panels on their personalized dashboard and add and remove them.
Global	Dashboard > Risk Header	/dashboard/risk_header/	Users can view the risk header at the top of the dashboard.

Permission Tab	Portal Permission	API Permission	Description
Global	Dashboard > Risk Header > Read	/dashboard/risk_header/read/	Users can view the risk header at the top of the dashboard.

Identity Providers

Table 33: Identity Providers Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Identity Providers	/identity_providers/	Users can view and modify the identity provider settings for identity providers.
Global	Identity Providers > Modify	/identity_providers/modify/	Users can modify the identity provider settings for identity providers.
Global	Identity Providers > Read	/identity_providers/read/	Users can view the identity provider settings for identity providers.

Metadata

Table 34: Certificate Metadata Types Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Metadata	/metadata/	Users can view and modify custom metadata attribute definitions.
Global	Metadata > Types	/metadata/types/	Users can view and modify custom metadata attribute definitions.
Global	Metadata > Types > Read	/metadata/types/read/	Users can view custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and the equivalent API functions.
Global	Metadata > Types > Modify	/metadata/types/modify/	Users can add, edit, and delete custom metadata attribute defin-

Permission Tab	Portal Permission	API Permission	Description
			itions on the Certificate Metadata page in the Management Portal and the equivalent API functions.

Monitoring

Table 35: Monitoring Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Monitoring	/monitoring/	Users can view, modify, and test the pending, issued, and denied certificate request alerts and the event handler registration settings.
Global	Monitoring > Handlers	/monitoring/handlers/	Users can view and modify the event handler registration settings.
Global	Monitoring > Handlers > Registration	/monitoring/handlers/registration/	Users can view and modify the event handler registration settings.
Global	Monitoring > Handlers > Registration > Modify	/monitoring/handlers/registration/modify/	Users can modify the event handler registration settings.
Global	Monitoring > Handlers > Registration > Read	/monitoring/handlers/registration/read/	Users can view the event handler registration settings.

Permission Tab	Portal Permission	API Permission	Description
Global	Monitoring > Alerts	/monitoring/alerts/	Users can view, modify, and test the pending, issued, and denied certificate request alerts.
Global	Monitoring > Alerts > Modify	/monitoring/alerts/modify/	Users can modify the pending, issued, and denied certificate request alerts, including the alert text, recipients, and event handlers. Users can also add new alerts, delete alerts, and configure the pending alert delivery schedule.
Global	Monitoring > Alerts > Read	/monitoring/alerts/read/	Users can view the pending, issued, and denied certificate request alerts.
Global	Monitoring > Alerts > Test	/monitoring/alerts/test/	Users can test the pending certificate request alerts, including sending email to recipients. Users must also have Read permissions for Alerts.

Privileged Access Management (PAM)

Table 36: Privileged Access Management Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Pam	/pam/	Users can view and modify any PAM provider.
Global	Pam > Modify	/pam/modify/	Users can add, edit, and delete any PAM provider.
PAM Provider	Pam > Modify	/pam/modify/#/	Users can add, edit, and delete the specified PAM provider.
Global	Pam > Read	/pam/read/	Users can view any PAM provider. Users can select any PAM providers to provide credentials within Keyfactor Command for: • Certificate Stores • Certificate Authorities • Workflow Definitions
PAM Provider	Pam > Read	/pam/read/#/	Users can view or select the specified PAM provider.

[Management] Portal

Table 37: Management Portal Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Portal	/portal/	Users can access the Management Portal.
Global	Portal > Read	/portal/read/	Users can access the Management Portal. This permission must be enabled for all roles that will access the Management Portal.

Reports

Table 38: Reports Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Reports	/reports/	Users can generate, view, and modify the delivery schedule for reports. Users can add, edit, and delete custom reports.
Global	Reports > Modify	/reports/modify/	Users can modify the delivery schedule for reports in Report Manager in the Management Portal and the equivalent API functions and add, edit, and delete custom reports. **Note:** Report scheduling is limited by collection permissions. Users in roles that have *Reports > Read and Modify* permissions will also need to have either global certificate *Read* permissions or *Read* collection permissions on individual collections to have the ability to add, edit, and delete schedules associated with collections. The user will not have access to add, edit, and delete schedules for any collections for which they do not have collection *Read* permissions in addition to *Reports* permissions if permissions are granted at a collection-by-collection level rather than globally.
Global	Reports > Read	/reports/read/	Users can generate and view reports.

Scripts

Table 39: Scripts Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Scripts	/scripts/	Users can view and modify scripts used in alert event handlers and workflows.
Global	Scripts > Modify	/scripts/modify/	Users can add, edit, and delete scripts used in alert event handlers and workflows.
Global	Scripts > Read	/scripts/read/	Users can view scripts used in alert event handlers and workflows.

Security

Note: Users can only edit security settings on security roles that are in a permission set to which they have been mapped. In some environments, there is only one permission set to which all users are mapped. Other environments may have implemented a more privileged access model with multiple permission sets. For more information on permission sets, see Permission Sets on page 1959.

Table 40: Security Settings Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Security	/security/	Users can view and modify the settings for Security Roles and Security Claims.
Global	Security > Modify	/security/modify/	Users can modify the settings for Security Roles and Security Claims.
Global	Security > Read	/security/read/	Users can view the settings for Security Roles and Security Claims. Users must also have the Read permission for System Settings to access this in the Management Portal.

SSH

See SSH Permissions on page 597 for additional information.

Table 41: SSH Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Ssh	/ssh/	Users can use all SSH functions.
Global	Ssh > Enterprise Admin	/ssh/enterprise_admin/	Users can use all SSH functions.
Global	Ssh > Server Admin	/ssh/server_admin/	Users can use all SSH functions, except creating server groups and assigning server group owners. Users have limited access to some functions based on server group ownership.
Global	Ssh > User	/ssh/user/	Users can generate their own SSH keys.

SSL

Table 42: SSL Management Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Ssl	/ssl/	Users can view and modify the SSL Discovery settings.
Global	Ssl > Modify	/ssl/modify/	Users can modify the SSL Discovery settings: • Create, edit, and delete networks, including scan schedules and notification recipients • Add, edit, and delete network ranges for networks • Add, edit, and delete agent pools • Add and remove discovered endpoints from monitoring
Global	Ssl > Read	/ssl/read/	Users can view the SSL Discovery pages in the Management Portal and the equivalent API functions, including defined networks and the network ranges configured for them, agent pools, and scan results. Users can use the query tool on the Results tab to find discovered endpoints and then view the discovered endpoints, including the details for the endpoints.

System Settings

Note: Users can only edit security settings on security roles that are in a permission set to which they have been mapped. In some environments, there is only one permission set to which all users are mapped. Other environments may have implemented a more privileged access model with multiple permission sets. For more information on permission sets, see Permission Sets on page 1959.

Table 43: System Settings Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	System Settings	/system_settings/	Users can modify the System Settings for: • Update SMTP Configuration for email delivery of reports and alerts • Installed components, including removing servers from use • Licensing, including the option to replace the existing license file
Global	System Settings > Modify	/system_settings/modify/	Users can modify the System Settings for: • Update SMTP Configuration for email delivery of reports and alerts • Installed components, including removing servers from use • Licensing, including the option to replace the existing license file
Global	System Settings > Read	/system_settings/read/	Users can view the System Settings for: • SMTP Configuration for email delivery of reports and alerts • Installed components • Licensing • General Alerts and Warnings about the health of the Keyfactor Command system (not related to a specific area of the product)

Workflows

Table 44: Workflows Security Role Permissions v2

Permission Tab	Portal Permission	API Permission	Description
Global	Workflows	/workflows/	Users can view and modify the configured workflow definitions and view and manage all initiated workflow instances.
Global	Workflows > Definitions	/workflows/definitions/	Users can view and modify the configured workflow definitions.
Global	Workflows > Definitions > Modify	/workflows/definitions/modify/	Users can modify both the built-in and any custom workflow definitions, including the name and description and the configuration for the steps. Users can also add new workflow definitions, delete workflow definitions, publish workflow definitions, and import and export workflow definitions.
Global	Workflows > Definitions > Read	/workflows/definitions/read/	Users can view the configured workflow definitions.
Global	Workflows > Instances	/workflows/instances/	Users can view and manage all initiated workflow instances.
Global	Workflows > Instances > Manage	/workflows/instances/manage/	Users can manage initiated workflow instances, including stopping, restarting, and deleting them.

Permission Tab	Portal Permission	API Permission	Description
Global	Workflows > Instances > Read	/workflows/instances/read/	Users can view all the workflow instances that have been initiated.
Global	Workflows > Instances > Read > Mine	/workflows/instances/read/mine/	Users can view the workflow instances that have been initiated by them (e.g. because they enrolled for a certificate).
Global	Workflows > Instances > Read > Pending	/workflows/instances/read/pending/	Users can view the workflow instances that have been initiated and are awaiting input from them. **Tip:** There is not a security permission at this level that controls whether users can provide input (a signal) to a workflow instance. This is controlled using the security roles configured on the specific workflow definition. Any user who holds one of the roles configured in the workflow step that

Permission Tab	Portal Permission	API Permission	Description
			requires a signal may provide the necessary input. The user does not need to hold the *Workflows > Instances > Read > Pending* permission in order to provide the input.

Version One Permission Model

The version one permission model was largely replaced in Keyfactor Command version 11.0, but is retained for backwards compatibility for use with select Keyfactor API endpoints.

Agent Auto-Registration

Table 45: Agent Auto-Registration Security Role Permissions v1

Portal Permission	API Permission	Description
Read	AgentAutoRegistration: *Read*	Users can view the orchestrator auto-registration settings; users must also have *Read* permissions for Agent Management to access this page in the Management Portal.
Modify	AgentAutoRegistration: *Modify*	Users can modify the orchestrator auto-registration settings.

Agent Management

Table 46: Agent Management Security Role Permissions v1

Portal Permission	API Permission	Description
Read	AgentManagement: *Read*	Users can: • View orchestrators, including filtering the Orchestrator Management grid • View orchestrator jobs, including status, schedules, failures and warnings
Modify	AgentManagement: *Modify*	Users can: • Manage orchestrators, including approving and disapproving them • Unschedule and reschedule orchestrator jobs

Alerts

Table 47: Alerts Security Role Permissions v1

Portal Permission	API Permission	Description
Read	WorkflowManagement: *Read*	Users can view the pending, issued, and denied certificate request alerts.
Modify	WorkflowManagement: *Modify*	Users can modify the pending, issued, and denied certificate request alerts, including the alert text, recipients, and event handlers. Users can also add new alerts, delete alerts, and configure the pending alert delivery schedule.
Test	WorkflowManagement: *Test*	Users can test the pending certificate request alerts, including sending email to recipients. Users must also have Read permissions for Alerts.

Application Settings

Table 48: Application Settings Security Role Permissions v1

Portal Permission	API Permission	Description
Read	ApplicationSettings: *Read*	Users can view the application settings.
Modify	ApplicationSettings: *Modify*	Users can modify the application settings.

Auditing

Table 49: Auditing Security Role Permissions v1

Portal Permission	API Permission	Description
Read	Auditing: *Read*	Users can access the Audit Log page in the Management Portal, and will be able to make API requests to obtain data from the audit log (query, etc.). The System Settings dropdown menu will display the Audit Log option to users with the **Auditing** Read permission.

Certificate Collections

Table 50: Certificate Collections Security Role Permissions v1

Portal Permission	API Permission	Description
Modify	CertificateCollections: *Modify*	Users can add or edit Certificate Collections. See Certificate Collection Permissions on page 627 for more information.

Certificate Enrollment

Table 51: Certificate Enrollment Security Role Permissions v1

Portal Permission	API Permission	Description
Enroll PFX	CertificateEnrollment: *EnrollPFX*	Users can use the PFX Enrollment page in the Management Portal and the equivalent API functions.
Enroll CSR	CertificateEnrollment: *EnrollCSR*	Users can use the CSR Enrollment page in the Management Portal and the equivalent API functions.
CSR Generation	CertificateEnrollment: *CsrGeneration*	Users can use the CSR Generation page in the Management Portal and the equivalent API functions.
Manage Pending CSRs	CertificateEnrollment: *PendingCsr*	Users can use the Pending CSRs page in the Management Portal and the equivalent API functions.

Certificate Metadata Types

Table 52: Certificate Metadata Types Security Role Permissions v1

Portal Permission	API Permission	Description
Read	CertificateMetadataTypes: *Read*	Users can read custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and the equivalent API functions.
Modify	CertificateMetadataTypes: *Modify*	Users can add, edit, and delete custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and the equivalent API functions.

Certificate Requests

Table 53: Certificate Requests Security Role Permissions v1

Portal Permission	API Permission	Description
Manage	WorkflowManagement: *Participate*	Users can participate in the pending, issued, and denied alerts by approving or denying certificate requests from the Certificate Requests page, from the individual pages reached from links included in alerts, or using the Keyfactor API /Workflow/Certificates endpoints. **Note:** In previous versions of Keyfactor Command, this permission was *Workflow Management: Participate*.

Certificate Store Management

Table 54: Certificate Store Management Security Role Permissions v1

See Container Permissions on page 629, Certificate Operations on page 45, Certificate Store Types on page 700 and Certificate Store Operations on page 413 for more information.

UI Permission	API Permission	Description
Read	CertificateStoreManagement: *Read*	Users can view the certificate stores and containers tabs on the *Locations* > *Certificate Stores* menu, and view certificate store types.

UI Permission	API Permission	Description
Schedule	CertificateStoreManagement: *Schedule*	Users can add certificates to certificate stores, renew/reissue certificates, schedule and remove certificates from certificate stores.
Modify	CertificateStoreManagement: *Modify*	Users can manage all operations regarding certificate stores—including the stores, containers, and discovery process—and certificate store types.

Certificates

Table 55: Certificates Security Role Permissions v1

Portal Permission	API Permission	Description
Read	Certificates: *Read*	Users can view certificates, including certificate history, and can download certificates. Users who also have Read permissions for Certificate Store Management or certificate store container permissions can add certificates to certificate stores from Certificate Search and Certificate Collections. The certificate operations possibly available to users with this permission are: • Add to Certificate Store (Also requires the *Read* and *Schedule Certificate Store Management* permissions) • Edit • Download • Get CSV • Identity Audit (Also requires the *Read Security Settings* permission) • Include Revoked checkbox • Include Expired checkbox • Renew (Also requires the *Read* and *Schedule Certificate Store Management* permissions) • Remove from Certificate Store (Also requires the *Read* and *Schedule Certificate Store Management* permissions) This permission can be applied at either the global or certificate collect level (see Certificate Collection Permissions on page 627. Users with **global Read** role permissions can browse to Certificate Search in the Management Portal and view all saved

Portal Permission	API Permission	Description
		certificate collections. They can view any certificate in the Keyfactor Command database and are not limited to just those returned by select collections. Users with this permission can view the certificates returned by searches and open the details of the certificates. Users with **collection-level Read** role permissions on a collection will see the collections to which they have been granted access appear on the Certificate Collections menu (if they have been configured to appear on the menu—see Certificate Collection Manager on page 85). The users will be able to view all the certificates in the collections and open the details of the certificates.
Edit Metadata	Certificates: *EditMetadata*	Users can modify certificate metadata for certificates in the Certificate Details on page 19 dialog (only information on the metadata tab can be edited) and the equivalent API functions. If the users have also been granted global *Read* permission on **Certificates**, they can modify the metadata of any certificates within the Keyfactor Command database. If the users have not been granted the global *Read* permission, they can only modify the certificates found in collections to which they have been granted collection-level *Read* access. **Note:** If you plan to edit metadata via the Keyfactor API, the user running the API needs only *Edit Metadata* permissions. *Read* permissions are not required.
Import	Certificates: *Import*	Users can import certificates using the Management Portal Add Certificate page or the Keyfactor API POST /Certificates/Import method. Users who also have Read permissions for Certificate Store Management or container permissions can add certificates to certificate stores from Add Certificate. **Note:** This permission cannot be applied at the certificate collection level.
Download with Private Key	Certificates: *Recover*	Users can download the certificates with their private key.

Portal Permission	API Permission	Description
Revoke	Certificates: *Revoke*	Users can revoke certificates through Keyfactor Command. Users with this role can use the revoke certificate operation on any certificates to which they have been granted access. This includes certificates that have been issued by a Microsoft or EJBCA CA configured for synchronization or by a cloud-based certificate vendor that is managed via a Keyfactor certificate gateway. **Important:** In order to successfully revoke certificates, the service account under which the Keyfactor Command application pool is running must be granted "Issue and Manage Certificates" and "Manage CA" permissions to the CA database as per Create Groups to Control Access to Keyfactor Command Features on page 2762, or, if delegation is configured for the CA, the user executing the revoke must have the "Issue and Manage Certificates" permissions while the application pool service account has the "Manage CA" permissions. If you are using explicit credentials to authenticate your CA (see Adding or Modifying a CA Record on page 354), it is the user specified on the CA configuration in Keyfactor Command who must have permissions on the CA.
Delete	Certificates: *Delete*	Users can delete certificates and, if applicable, the private keys of the certificates from the Keyfactor Command database.
Import Private Key	Certificates: *ImportPrivateKey*	Users can save the private key for the certificate in the Keyfactor Command database. Users with this role can add a certificate with an associated private key through the Add Certificate option under the Certificate Locations menu (see Add Certificate on page 74) and the private key will be stored in the Keyfactor Command database. Users must also be granted the *Import* role in order to be able to use the Add Certificate feature. **Note:** This permission cannot be applied at the certificate collection level.

Dashboard

Table 56: Dashboard Security Role Permissions v1

Portal Permission	API Permission	Description
Read	Dashboard: *Read*	Users can view the panels on their personalized dashboard and add and remove them.
Risk Header	Dashboard: *RiskHeader*	Users can view the risk header at the top of the dashboard.

Event Handler Registration

Table 57: Event Handler Registration Security Role Permissions v1

Portal Permission	API Permission	Description
Read	EventHandlerRegistration: *Read*	Users can view the event handler registration settings.
Modify	EventHandlerRegistration: *Modify*	Users can modify the event handler registration settings.

Identity Providers

Table 58: Identity Providers Security Role Permissions v1

Portal Permission	API Permission	Description
Read	IdentityProviders: *Read*	Users can view the identity provider settings.
Modify	IdentityProviders: *Modify*	Users can modify the identity provider settings.

Mac Auto-Enroll Management

Table 59: Mac Auto-Enroll Management Security Role Permissions v1

Portal Permission	API Permission	Description
Read	MacAutoEnrollManagement: *Read*	Users can view the Mac Auto-Enroll Management settings.
Modify	MacAutoEnrollManagement: *Modify*	Users can modify the Mac Auto-Enroll Management settings.

Management Portal

Table 60: Management Portal Security Role Permissions v1

Portal Permission	API Permission	Description
Read	AdminPortal: *Read*	Users can access the Management Portal. This permission must be enabled for all roles that will access the Management Portal.

Monitoring

Table 61: Monitoring Security Role Permissions v1

Portal Permission	API Permission	Description
Read	Monitoring: *Read*	Users can view the expiration alerts in the Certificate Alerts in the Management Portal and the equivalent API functions, including the alert schedule.
Modify	Monitoring: *Modify*	Users can modify the expiration alerts, including the alert text, recipients and event handlers. Users can also add new alerts, delete alerts and configure the expiration alert delivery schedule.
Test	Monitoring: *Test*	Users can test the expiration alerts, including sending email to recipients. Users must also have Read permissions for Monitoring to access this in the Management Portal.

PKI Management

Table 62: PKI Management Security Role Permissions v1

Portal Permission	API Permission	Description
Read	PkiManagement: *Read*	Users can view PKI management settings within: • Certificate Authorities • Certificate Templates • Revocation Monitoring
Modify	PkiManagement: *Modify*	Users can modify PKI management settings to: • Import, add, edit, and delete certificate authorities • Import and edit certificate templates • Add, edit, delete, and test revocation monitoring endpoints

Portal Permission	API Permission	Description
		• Configure revocation monitoring schedules • Configure revocation monitoring recipients

Privileged Access Management

Table 63: Privileged Access Management Security Role Permissions v1

Portal Permission	API Permission	Description
Read	PrivilegedAccessManagement: *Read*	Users can view PAM providers.
Modify	PrivilegedAccessManagement: *Modify*	Users can add, edit, and delete PAM providers.

Reports

Table 64: Reports Security Role Permissions v1

Portal Permission	API Permission	Description
Read	Reports: *Read*	Users can generate and view reports.
Modify	Reports: *Modify*	Users can modify the delivery schedule for reports in Report Manager in the Management Portal and the equivalent API functions and add, edit, and delete custom reports. **Note:** Report scheduling is limited by collection permissions. Users in roles that have *Reports: Read and Modify* permissions will also need to have *Read* collection permissions on individual collections to have the ability to add, edit, and delete schedules associated with collections. The user will not have access to add, edit, and delete schedules for any collections for which they do not have collection *Read* permissions in addition to *Reports* permissions.

Scripts

Table 65: Scripts Security Role Permissions v1

Portal Permission	API Permission	Description
Read	Scripts: *Read*	Users can view scripts.
Modify	Scripts: *Modify*	Users can add, edit, and delete scripts.

Security Settings

Table 66: Security Settings Security Role Permissions v1

Portal Permission	API Permission	Description
Read	SecuritySettings: *Read*	Users can view the settings for Security Roles and Security Claims. Users must also have the Read permission for System Settings to access this in the Management Portal.
Modify	SecuritySettings: *Modify*	Users can modify the settings for Security Roles and Security Claims.

SSH

Table 67: SSH Security Role Permissions v1

Portal Permission	API Permission	Description
User	SSH: *User*	Users can generate their own SSH keys.
Server Admin	SSH: *ServerAdmin*	Users can use all SSH functions, except creating server groups and assigning server group owners. Users have limited access to some functions based on server group ownership (see SSH Permissions on page 597).
Enterprise Admin	SSH: *EnterpriseAdmin*	Users can use all SSH functions (see SSH Permissions on page 597).

SSL Management

Table 68: SSL Management Security Role Permissions v1

Portal Permission	API Permission	Description
Read	SslManagement: *Read*	Users can view the SSL Discovery pages in the Management Portal and the equivalent API functions, including defined networks and the network ranges configured for them, agent pools, and scan results. Users can use the query tool on the Results tab to find discovered endpoints and then view the discovered endpoints, including the details for the endpoints.
Modify	SslManagement: *Modify*	Users can modify the SSL Discovery settings: • Create, edit, and delete networks, including scan schedules and notification recipients • Add, edit, and delete network ranges for networks • Add, edit, and delete agent pools • Add and remove discovered endpoints from monitoring

System Settings

Table 69: System Settings Security Role Permissions v1

Portal Permission	API Permission	Description
Read	SystemSettings: *Read*	Users can view the orchestrator auto-registration settings; users must also have *Read* permissions for Agent Management to access this in the Management Portal. Users can view the System Settings for: • SMTP Configuration for email delivery of reports and alerts • Installed components • Licensing • General Alerts and Warnings about the health of the Keyfactor Command system (not related to a specific area of the product)
Modify	SystemSettings: *Modify*	Users can modify the System Settings for: • Update SMTP Configuration for email delivery of reports and alerts • Installed components, including removing servers from use • Licensing, including the option to replace the existing license file

Workflow Definitions

Table 70: Workflow Definitions Security Role Permissions v1

Portal Permission	API Permission	Description
Read	WorkflowDefinitions: *Read*	Users can view the configured workflow definitions.
Modify	WorkflowDefinitions: *Modify*	Users can modify both the built-in and any custom workflow definitions, including the name and description and the configuration for the steps. Users can also add new workflow definitions, delete workflow definitions, publish workflow definitions, and import and export workflow definitions.

Workflow Instances

Table 71: Workflow Instances Security Role Permissions v1

Portal Permission	API Permission	Description
ReadAll	WorkflowInstances: *ReadAll*	Users can view all the workflow instances that have been initiated.
Read - Assigned To Me	WorkflowInstances: *ReadAssignedToMe*	Users can view the workflow instances that have been initiated and are awaiting input from them. **Tip:** There is not a security permission at this level that controls whether users can provide input (a signal) to a workflow instance. This is controlled using the security roles configured on the specific workflow definition. Any user who holds one of the roles configured in the workflow step that requires a signal may provide the necessary input. The user does not need to hold the *Read - Assigned To Me Workflow Instances* permission in order to provide the input.
Read - Started By Me	WorkflowInstances: *ReadMy*	Users can view the workflow instances that have been initiated by them (e.g. because they enrolled for a certificate).
Manage	WorkflowInstances: *Manage*	Users can manage initiated workflow instances, including stopping, restarting, and deleting them.

Adding or Modifying a Security Role

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Security > Modify
Security > Read

To add a new security role or update an existing role:

1. In the Management Portal, browse to *System Settings Icon* > *Security Roles and Claims.*
2. On the Security Roles and Claims page, select the Security Role tab and click **Add** from the menu at the top of the grid to add a new security role, or highlight a row and click **Edit** from the top of the grid or from the right click menu to modify an existing role.

   Note: The Administrator and Reporting API Access roles cannot be edited or deleted—they are shown as *immutable* in the grid.

3. Either the **Add Security Role** dialog or **Role information For <role>** dialog will open. Fill in each tab of the dialog with the information desired for the selected security role as described below.

Details Tab

On the details tab, give the role a **Name** and **Description**. Both fields are required. In the **Permission Set** dropdown, select a permission set to apply to the role. For more information about permission sets, see Permission Sets on page 1959.

Note: Selecting a permission set limits the permissions available for selection in the dialog to those supported by the permission set. If you select a permission set other than the Global permission set, some permissions may not appear in the dialogs. This includes the Collection Permissions, Container Permissions and PAM Provider Permissions tabs, which only appear if the permissions for these are included in the selected permission set.
If you change the permission set for a role after permissions have already been granted for the role, any permissions not contained in the new permission set will be removed from the role upon save.

Global Permissions Tab

On the Global Permissions tab, check the boxes for the permissions that are appropriate for the new role (see Security Role Permissions on page 632).

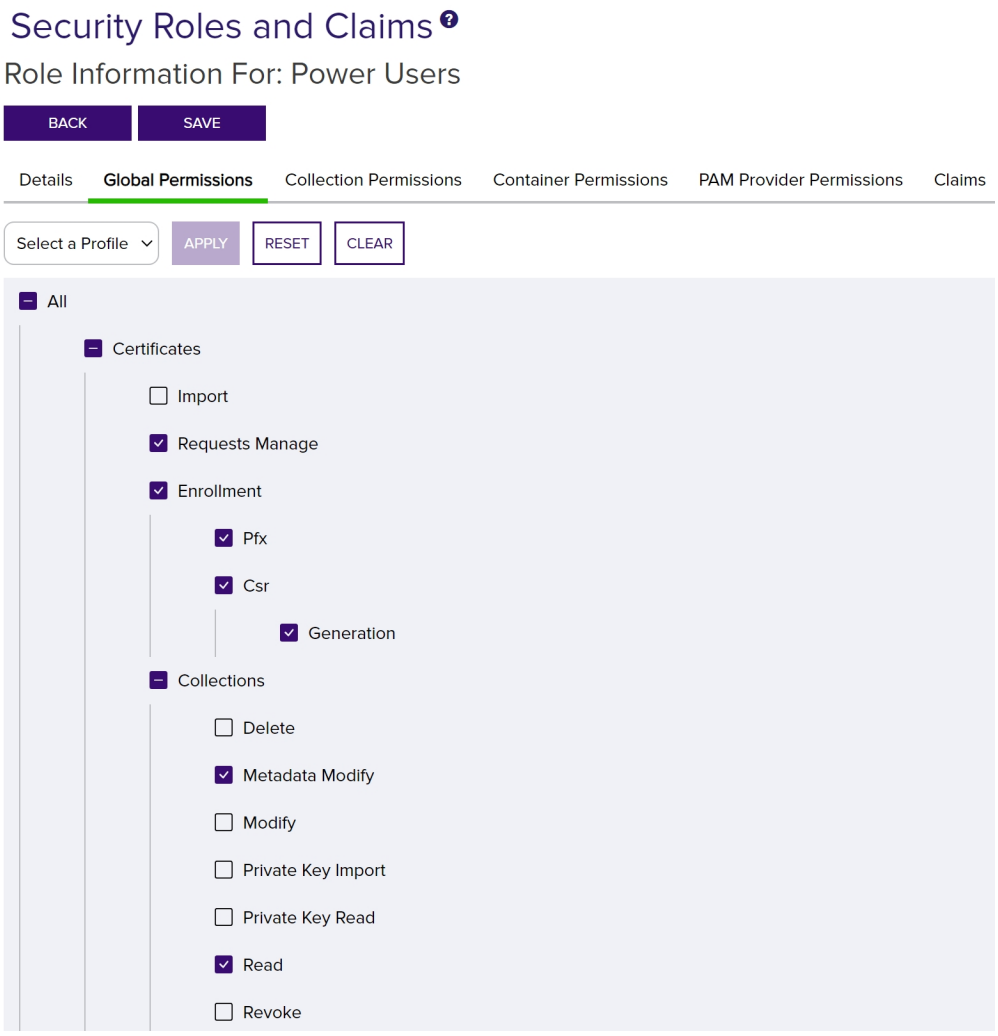


Figure 351: Grant Global Permissions to a Security Role

Tip: If desired, use the dropdown at the top to enable all the **Read** boxes (*Read Only*) or **All** the boxes (*Select All*). Click **Apply** to apply the selection in the dropdown across all permissions. Click **Reset** to return the dialog to the state it was in when last saved and remove any changes made since opening the permissions for editing. Click **Clear** to uncheck all the boxes.

Note: For the most part, when you grant Modify role permissions to an area in the Management Portal, you must also grant Read role permissions to that same area for

that security role to receive full functionality. Granting Modify without Read to a user or a group can result in unexpected behavior. See also Certificate Collection Permissions on page 627.

Note: Security roles for SSH key management are structured somewhat differently than those for most of the rest of the product set, as they don't use the standard Read and Modify convention. For more information, see SSH Permissions on page 597.

Collection Permissions Tab

Optionally, on the Collection Permissions tab, highlight each certificate collection you would like to set permissions for and click the toggle button for each desired permission (see Certificate Collection Permissions on page 627). If you do not opt to set permissions on any collections, the permissions set on the Global Permissions tab will apply to all collections.

Along the top of the Collection Permissions tab, you can see the system-wide certificate collection permissions that have been configured. This is useful for reference. The system-wide settings may also be edited from this tab, if desired.

Any permissions that have been enabled at the system-wide level will not be available for configuration at the collection-level. Toggles for these will be grayed out.

Tip: The search bar at the top of Collection Name column on the collections tab can make it easier to find collections if you have a large number of them.

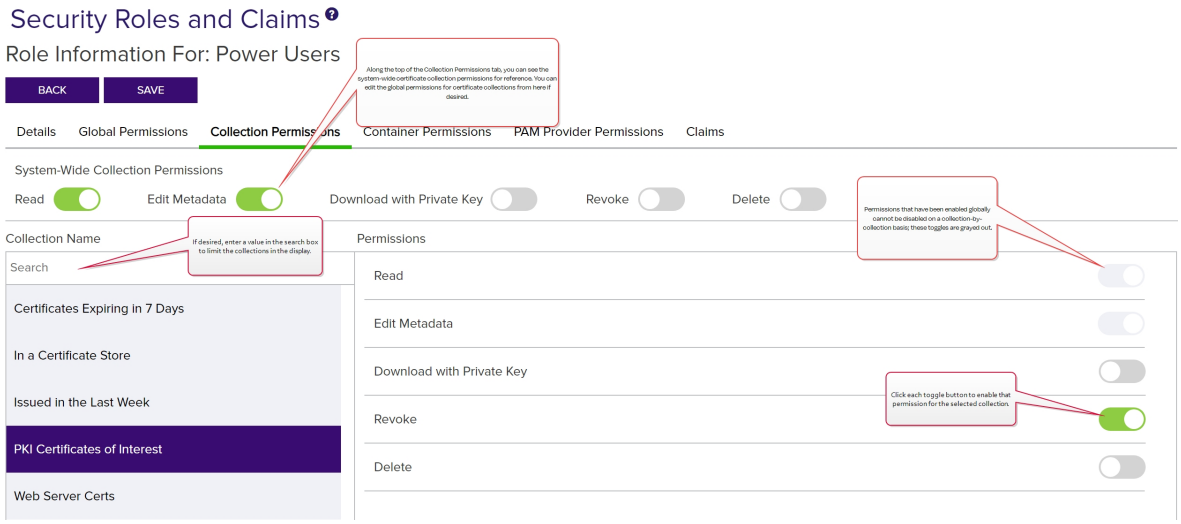


Figure 352: Grant Collection Permissions to a Security Role

Container Permissions Tab

Optionally, on the Container Permissions tab, highlight each certificate store container you would like to set permissions for and click the toggle button for each desired permission (see Container Permissions on page 629). If you do not opt to set permissions on any certificate store containers, the permissions set on the Global Permissions tab will apply to all certificate store containers.

> **Tip:** The search bar at the top of Container Name column on the containers tab can make it easier to find certificate store containers if you have a large number of them.

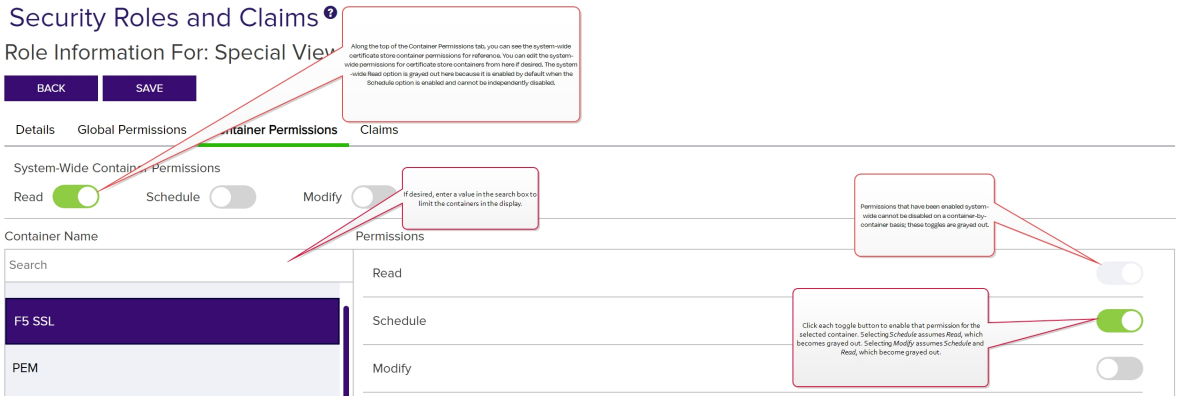


Figure 353: Grant Container Permissions to a Security Role

PAM Provider Permissions Tab

Optionally, on the PAM Provider Permissions tab, highlight each PAM provider you would like to set permissions for and click the toggle button for each desired permission (see PAM Permissions on page 631). If you do not opt to set permissions on any PAM providers, the permissions set on the Global Permissions tab will apply to all PAM providers.

> **Tip:** The search bar at the top of PAM Provider Name column on the PAM provider permissions tab can make it easier to find PAM providers if you have a large number of them.

Claims Tab

On the Claims tab, you associate security claims with the security role to grant the permissions associated with the role to claim owners. You can associate existing security claims or add new security claims and associate them in one step. Claim to role associations can also be removed from this tab, but claims cannot be deleted from here.

To add a new claim and associate it with the role in one step:

a. On the Claims tab, click **Add New** from the menu at the top of the grid to add a new security claim and associate it.

b. In the Add Claim dialog, select a **Claim Type** in the dropdown. Supported claim types are:

   - Active Directory User

     Active Directory user account.
   - Active Directory Group
   - Active Directory group.
   - Active Directory Computer
   - Active Directory machine account.
   - OAuth Client Id

     An open authorization client application claim.
   - OAuth Object Id

     An open authorization claim of a type not covered by client, role or subject.
   - OAuth Role

     An open authorization group claim.
   - OAuth Subject

     An open authorization user claim.

c. In the **Claim Value** field, enter the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$). For an identity provider other than Active Directory, users are referenced by GUID (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716) and groups (roles) are referenced by role name.

   Important: The built-in Active Directory groups Domain Admins and Enterprise Admins cannot be used directly to grant access to the Management Portal due to how these groups function within Windows. You can create a custom Active Directory group, reference that group in the Management Portal, and add the built-in Domain Admins or Enterprise Admins group to that custom group, if desired.

   Tip: If you're using a federated identity provider with Keyfactor Identity Provider to authenticate your users, enter the Keyfactor Identity Provider identity for that user

or a role in Keyfactor Identity Provider that user has been granted (see Federating from Keyfactor Identity Provider on page 2732).

d. In the **Provider** dropdown, select the correct identity provider for the claim—either Active Directory or the identity provider name you configured using the configuration wizard.

> **Tip:** If you're using a federated identity provider with Keyfactor Identity Provider to authenticate your users, select the Keyfactor Identity Provider name in the drop-down.

e. In the **Description** field, enter a description for the security claim. For example, the user's full name or the group owner. This field is required.

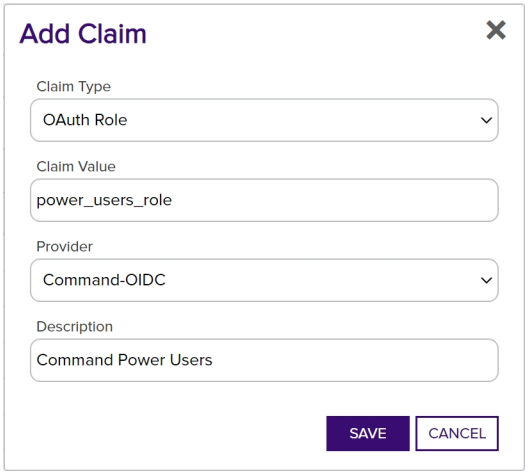


Figure 354: Add a Security Claim for an OAuth Identity Provider Role

To associate an existing claim with the role:

a. On the Claims tab, highlight a row and click **Add Existing** from the top of the grid or from the right click menu to associate an existing claim.

b. On the Select Existing Claims dialog, check the box next to each desired claim and click **Include** or **Include and Close** to associate the claim(s) with this role. You may use the search at the top of the dialog to limit the claims shown in the dialog grid (see Using the Security Claim Search Feature on page 698).

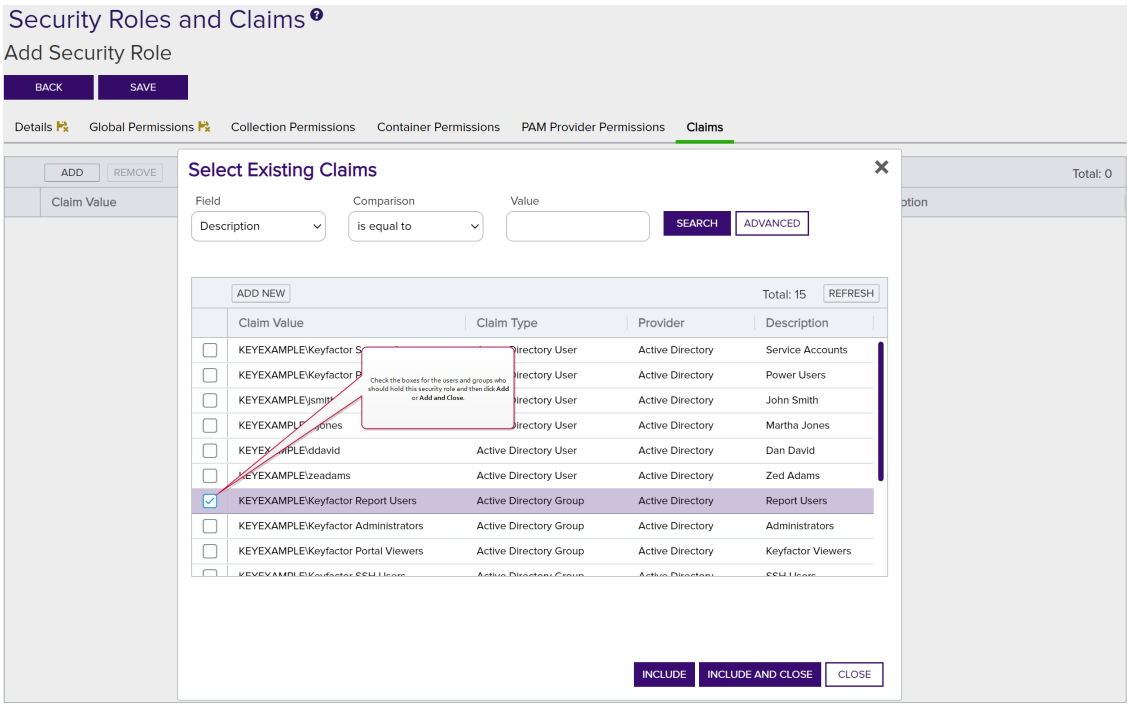


Figure 355: Associate Existing Claims with a Security Role

To remove the association of a claim to the role, on the Claims tab, highlight one or more rows and click **Remove** from the top of the grid or select a single row and choose **Remove** from the right click menu.

4. Click **Save** to save the role.

Copying a Security Role

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Security > Modify
Security > Read

To copy a security role:

1. In the Management Portal, browse to *System Settings Icon* > *Security Roles and Claims.*
2. On the Security Roles and Claims page, select the Security Role tab. Highlight a row and click **Copy** from the top of the grid or from the right click menu to copy an existing role.
3. Click OK to the Confirm Operation message.

Note: Copying a security role will also assign the new role to all the same security claims as the original role.

4. The name will automatically be set to *Copy of (original role name)* with the same description as the original role. Update the name and description and click **Save**.

Note: The Administrators and Reporting API Access roles cannot be copied.

Deleting a Security Role

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Security > Modify
Security > Read

To delete a security role:

1. In the Management Portal, browse to *System Settings Icon* > *Security Roles and Claims*.
2. On the Security Roles and Claims page, select the Security Role tab. Highlight a row and click **Delete** from the top of the grid or from the right click menu to delete an existing role.

Note: The Administrators and Reporting API Access roles cannot be edited or deleted.

Tip: You can view all the permissions set for a given role at a glance by granting *one* role to *one* claim only (and no other roles) and then using the View Permissions option for the claim (see Viewing Permissions for a Security Claim on page 695).

Using the Security Role Search Feature

Note: The security role search skips the validation check when loading for improved performance. The validation still occurs when loading a single record, so users will encounter an error when trying to work with an invalid role.

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

Name	ClaimProviderName
Complete or partial matches with the name of the security role.	Complete or partial matches with the name of the claim provider (e.g. Active Directory) for claims associated with the roles.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.

- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

Security Claim Operations

From the *Claims* tab of the Security Roles and Claims page in Keyfactor Command you can create the individual claims that will be associated with one or more security roles to define the user access to Keyfactor Command. Prior to adding new claims, it is recommended that you create all of the security roles you require (see Security Role Operations on page 683) so they can be assigned to the new claims.

Tip: To associate a security claim with a role, edit the desired security role, and select claims on the Claims tab (see Claims Tab on page 686).

Adding or Modifying Security Claims

Security claims can be added on the Security Roles and Claims page Claims tab and also from the Claims tab within each role when editing roles on the Security Roles and Claims page Roles tab.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Security > Read
Security > Modify

To add a new security claim or update an existing claim:

1. In the Management Portal, browse to *System Settings Icon* > *Security Roles and Claims.*
2. On the Security Roles and Claims page, select the Claims tab and click **Add** from the menu at the top of the grid to add a new security claim, or highlight a row and click **Edit** from the top of the grid or from the right click menu to modify an existing claim.
3. In the Add/Edit Claim dialog, select a **Claim Type** in the dropdown. Supported claim types are:

    - Active Directory User

      Active Directory user account.
    - Active Directory Group
    - Active Directory group.
    - Active Directory Computer
    - Active Directory machine account.
    - OAuth Client Id

      An open authorization client application claim.
    - OAuth Object Id

      An open authorization claim of a type not covered by client, role or subject.
    - OAuth Role

      An open authorization group claim.
    - OAuth Subject

      An open authorization user claim.

    Note: This field cannot be modified on an edit.

4. In the **Claim Value** field, enter the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$). For an identity provider other than Active Directory, users are referenced by GUID (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716) and groups (roles) are referenced by role name.

Important: The built-in Active Directory groups Domain Admins and Enterprise Admins cannot be used directly to grant access to the Management Portal due to how these groups function within Windows. You can create a custom Active Directory group, reference that group in the Management Portal, and add the built-in Domain Admins or Enterprise Admins group to that custom group, if desired.

Tip: If you're using a federated identity provider with Keyfactor Identity Provider to authenticate your users, enter the Keyfactor Identity Provider identity for that user or a role in Keyfactor Identity Provider that user has been granted (see Federating from Keyfactor Identity Provider on page 2732).

Note: This field cannot be modified on an edit.

5. In the **Provider** dropdown, select the correct identity provider for the claim—either Active Directory or the identity provider name you configured using the configuration wizard.

   Tip: If you're using a federated identity provider with Keyfactor Identity Provider to authenticate your users, select the Keyfactor Identity Provider name in the dropdown.

   Note: This field cannot be modified on an edit.

6. In the **Description** field, enter a description for the security claim. For example, the user's full name or the group owner. This field is required.

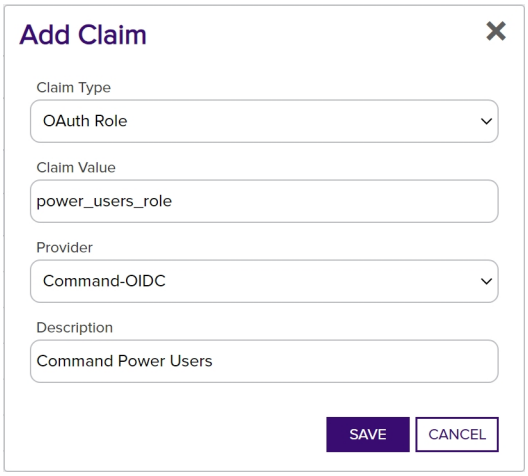


Figure 356: Add a Security Claim for an OAuth Identity Provider Role

Deleting a Security Claim

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Security > Read
Security > Modify

To delete a security claim:

1. In the Management Portal, browse to *System Settings Icon* > *Security Roles and Claims.*
2. Select the **Claims tab** of the page. Highlight the claim you want to delete and click **Delete** at the top of the grid. Or right-click the row in the grid and choose **Delete** from the right-click menu.

Important: Do not delete the last claim associated with the Administrator role or you will lose access to the administrative features of the Management Portal.

Viewing Permissions for a Security Claim

Within this dialog you can view the global permissions for the claim, certificate store container permissions, or certificate collection permissions.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Security > Read

1. In the Management Portal, browse to *System Settings Icon* > *Security Roles and Claims.*
2. Select the **Claims tab** of the page. Highlight the claim you want to view permissions for and click **View Permissions** at the top of the grid. Or right-click the row in the grid and choose **View Permissions** from the right-click menu.

Note: The **Global Permissions** tab of the *View Permissions* dialog will be updated in a future release to match the new Keyfactor Command v11 security structure. Currently it is showing the old structure which is not a one-to-one match to the v11 structure. The other tabs are up-to-date and accurate.

If the user or group has been granted more than one role, you see the permissions of all the roles granted to the user or group consolidated together on the **View Permissions** dialog for easy viewing. Hover over a specific permission to see how that permission was granted.

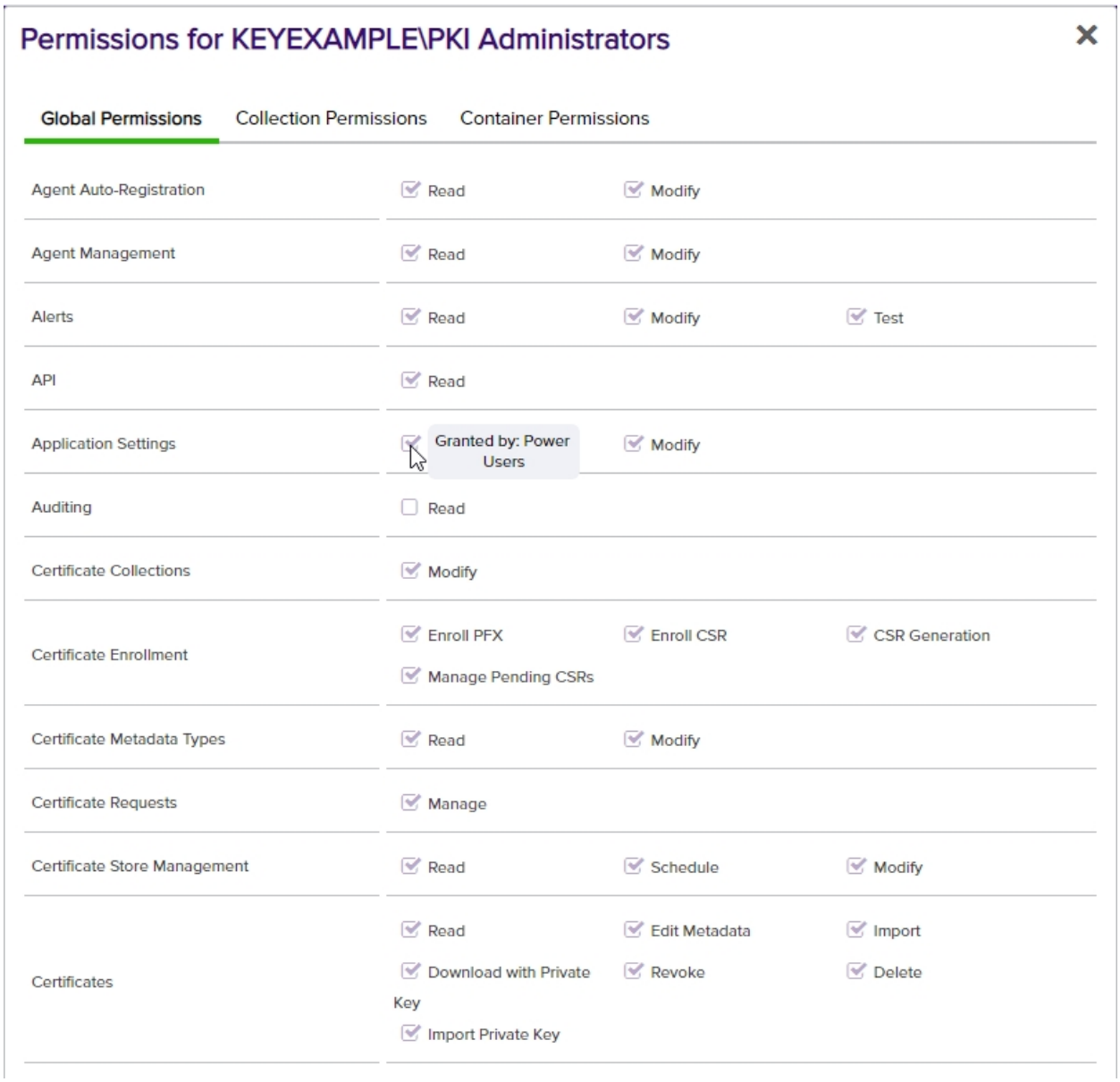


Figure 357: View Global Permissions for a Security Claim

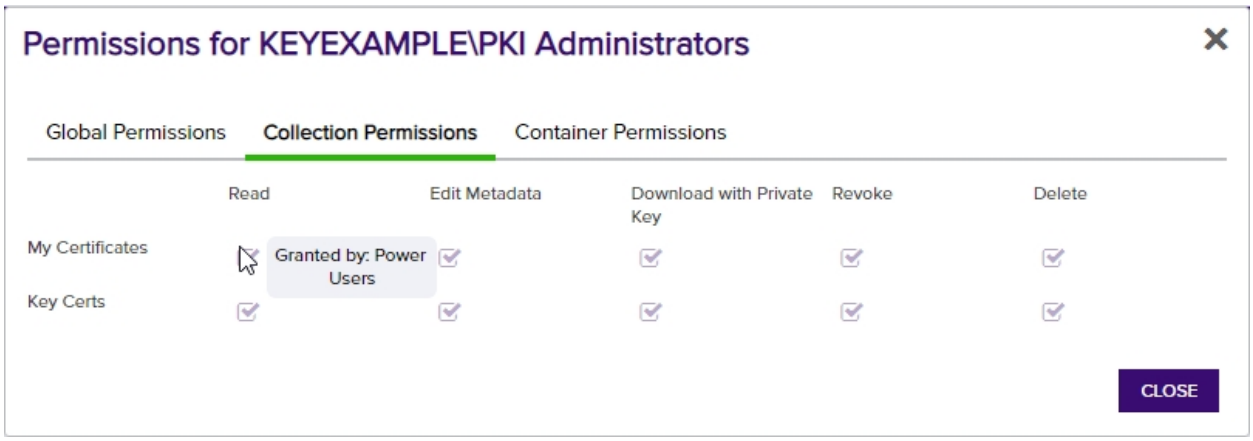


Figure 358: Collection Permissions for a Security Claim

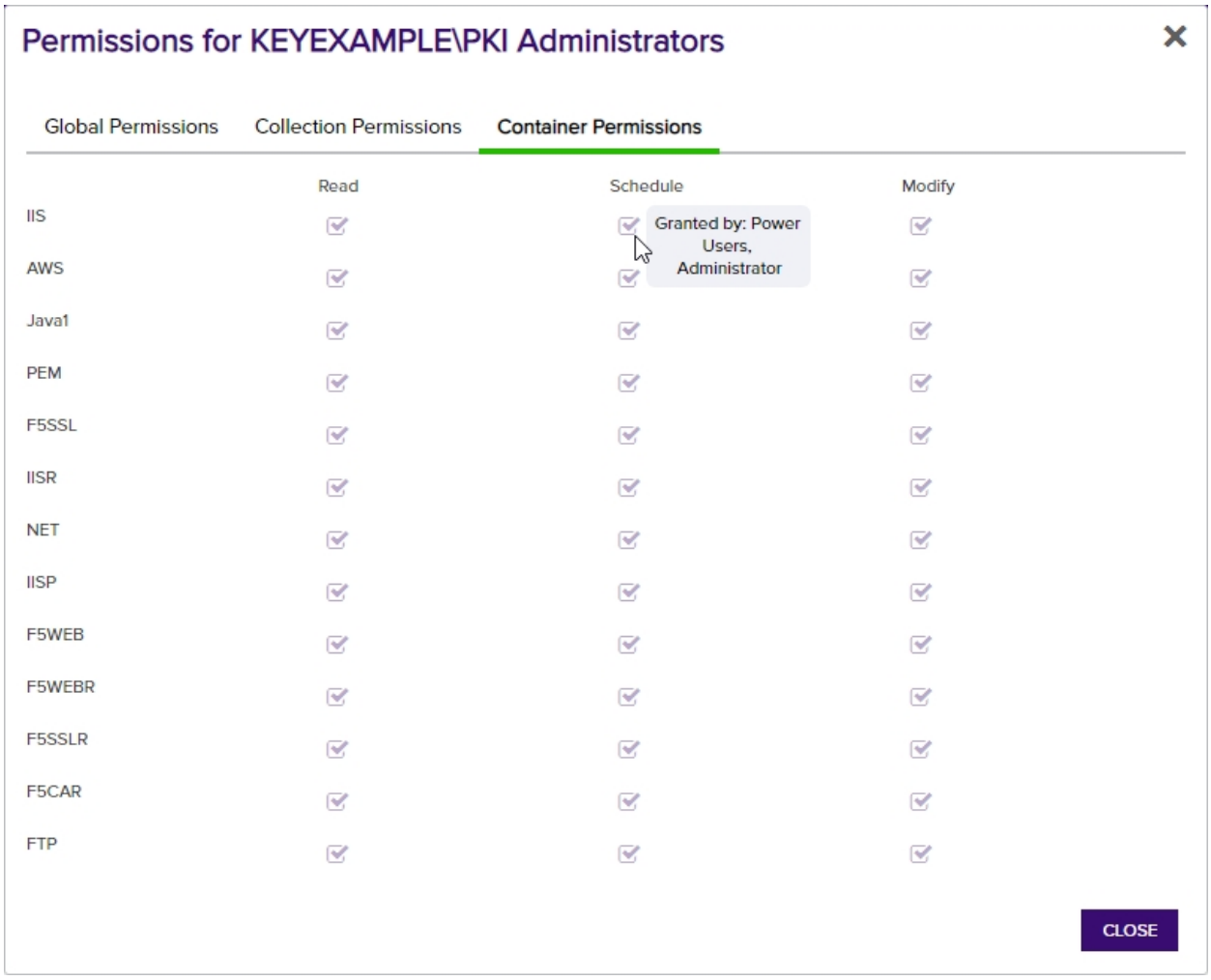


Figure 359: Container Permissions for a Security Claim

Viewing Roles for a Security Claim

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Security > Read

To view the roles associated with a claim:

1. In the Management Portal, browse to *System Settings Icon* > *Security Roles and Claims*.
2. Select the **Claims tab** of the page. Highlight the claim you want to view roles for and click **View Roles** at the top of the grid. Or right-click the row in the grid and choose **View Roles** from the right-click menu.

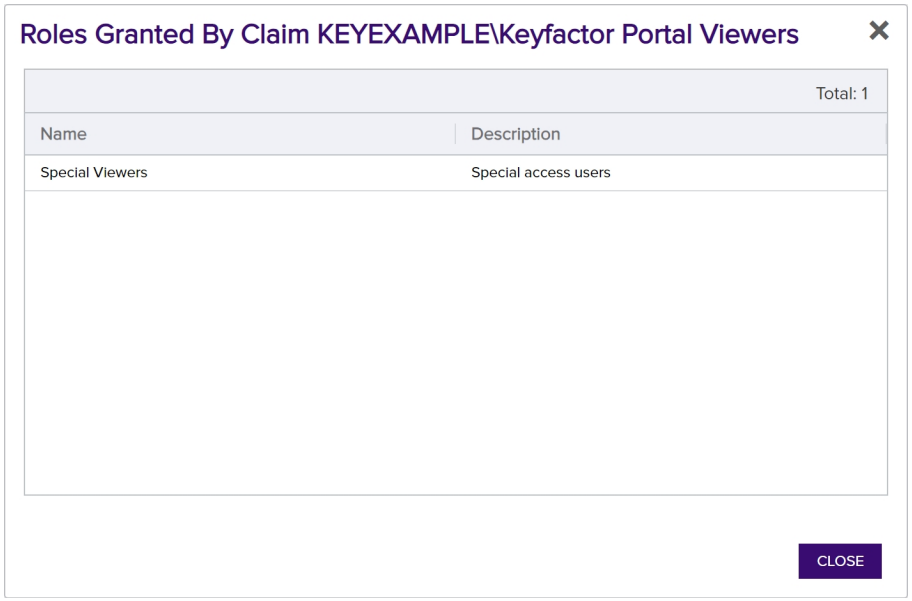


Figure 360: View Roles associated with a Security Claim

Using the Security Claim Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

ADClaimValue

Complete matches with the name of the security claim for Active Directory (e.g. KEYEXAMPLE\Keyfactor Administrators).

ClaimType

The claim type matches or doesn't match the referenced value. Supported claim types are:

- Computer
- Group
- User
- OAuthClientId
- OAuthOid
- OAuthRole
- OAuthSubject

ClaimValue

Complete or partial matches with the name of the security claim for OAuth providers (e.g. Keyfactor Administrators).

Description

Complete or partial matches with the description for the security claim.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

2.1.11.3 Certificate Store Types

Certificate store types allow you to define types of locations to contain certificates. These locations can be defined for operations such as inventory, management, discovery, and reenrollment.

Custom certificate store types can be created for use with custom extensions (see Certificate Store Type Operations on the next page).

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Certificate Store Type Operations

Certificate store types define locations against which Keyfactor Command can perform predefined operations. New ones are commonly added for custom orchestrators created with the Keyfactor AnyAgent, the Keyfactor Native Agent, or another of the tools in the Keyfactor Integration SDK (see Orchestrators on page 481).

The certificate store types page displays a list of the currently defined types and offers the options to create new types, edit existing types and delete types. It is not possible to update built-in certificate store types because doing so will break the associated orchestrator functionality.

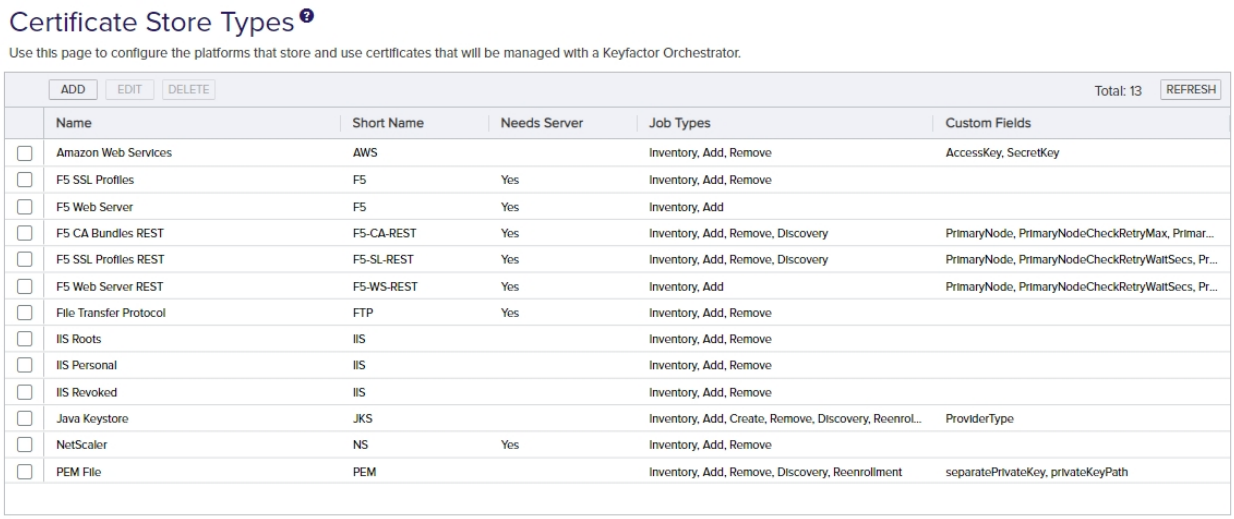


Figure 361: Certificate Store Types

Adding or Editing a Certificate Store Type

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Modify
Certificate Stores > Read

Note: Certificate store types can only be updated in a very limited way if they are actively in use (there are any certificate stores defined for them). Updates to the Name and Short Name

are supported in this case as are additions to the Supported Job Types, but no other updates can be saved.

To create or modify a certificate store type:

1. In the Management Portal, browse to *System Settings Icon* > *Certificate Store Types*.
2. On the Certificate Store Types page, click **Add** to create a new certificate store type, or click **Edit** from either the top or right-click menu to modify an existing one.
3. In the Certificate Store Types dialog, you will see four tabs. Complete the dialog with appropriate information using the following information:

Basic Tab

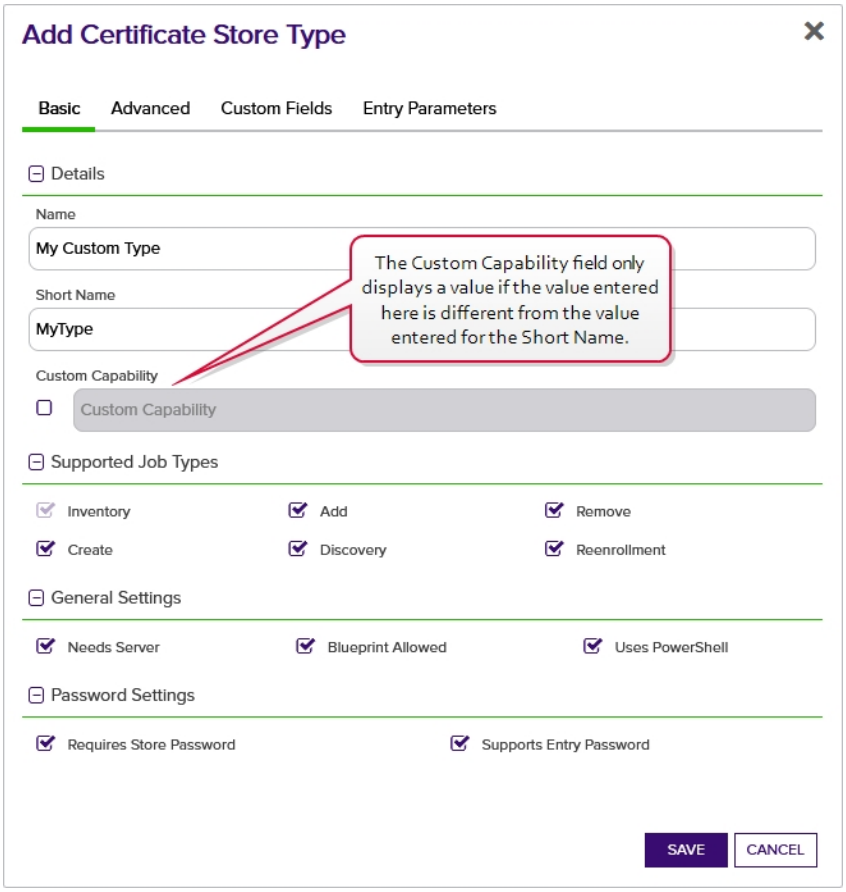


Figure 362: Add New Certificate Store Type: Basic Tab

- **Name:** Enter a user friendly recognizable name for the certificate store type.
- **Short Name:** Enter a short name identifier for the certificate store type. This value is used by the Keyfactor Universal Orchestrator installation and configuration tools to validate the orchestrator capabilities.
- **Custom Capability:** If desired, check this box to allow you to define a custom capability name. By default, the Short Name is used as the capability name, and in most cases a separate capability name is not needed. The capability name you set here corresponds to configurations made in the manifest.json file for your custom orchestrator extension.

  Tip: This box shows as checked only if the value entered in the *Custom Capability* does not match the value entered in the *Short Name*. If you check the box, enter a value that matches the short name value, save the record and open it again, the box will show unchecked and the *Custom Capability* field will show empty since the value matches the *Short Name* value.

  Note: The *Custom Capability* cannot be changed on an edit if an orchestrator has registered with Keyfactor Command, been approved, and included the certificate store type in its capability list. If you change the *Short Name* in this circumstance, the *Custom Capability* box will be checked and the value set to the original value of the *Short Name*.

- **Supported Job Types**
  Select the job capabilities required to support the store type.
  - **Inventory:** Determine what is in the certificate store(s) and report the contents to Keyfactor Command. This capability is required for all store types.
  - **Add:** Add new certificates to a certificate store.
  - **Remove:** Remove certificates from a certificate store.
  - **Create:** Create a new certificate store.
  - **Discovery:** Determine what certificate stores of this type are on the device.
  - **Reenrollment:** Generate a keypair on the device and submit a certificate signing request using on-device key generation (ODKG).
- **General Settings**
  - **Needs Server:** Select if server access is required for adding certificate stores to the certificate store type. If selected, a user will be prompted for a username and password to connect to the remote server.

- **Blueprint Allowed:** Select whether certificate stores of this type will be included when creating or applying blueprints. For more details, see Generating and Applying Blueprints on page 501.
- **Uses PowerShell:** Select if the jobs for this store type are implemented by PowerShell instead of a .NET class.

- **Password Settings**
  - **Requires Store Password:** Select to mandate that a password be entered and authenticated when creating stores of this type. This password secures the store as a whole.
  - **Supports Entry Password:** Select to allow an entry password to be entered and authenticated when adding a certificate to a store. This password secures a single certificate within the store.

Advanced Tab

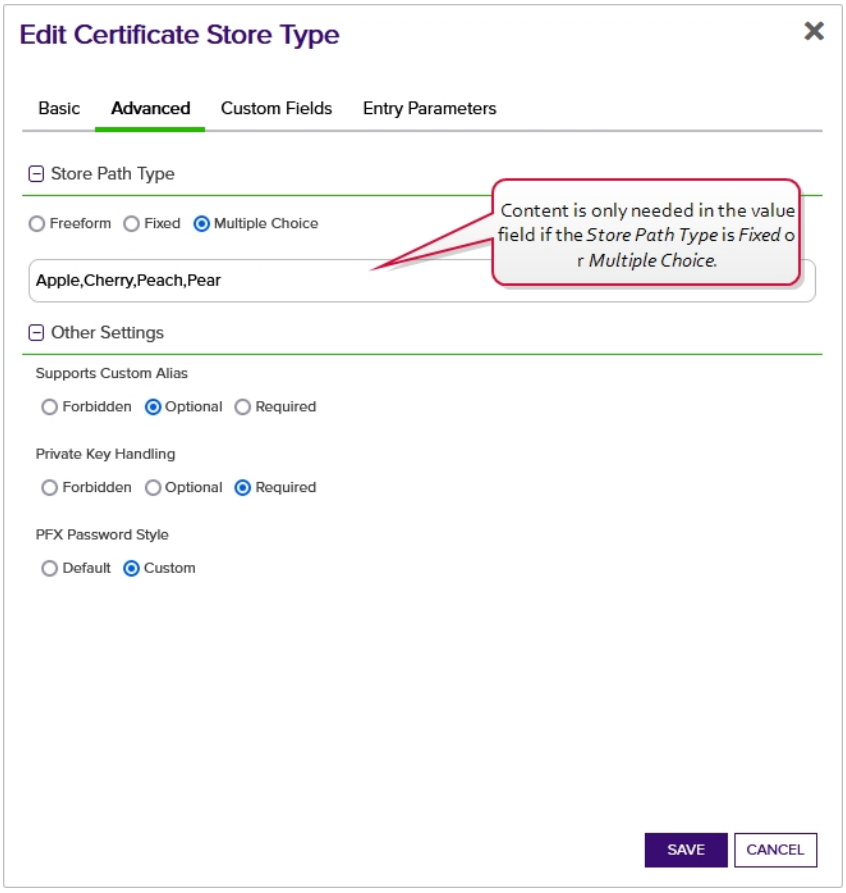


Figure 363: Add New Certificate Store Type: Advanced Tab

- **Store Path Type:**
  - **Freeform:** Select if users are required to enter a path defining the store location.
  - **Fixed:** Select if a store path does not apply, generally one store per device (e.g. IIS).
  - **Multiple Choice:** Select to allow users to select an option during certificate store creation.

  If *Store Path Type* is *Fixed* or *Multiple Choice*, a value should be provided in the value field. For multiple choice, this should be a comma separated list of values that users will be able to select from when defining a certificate store location.
- Other Settings
  - **Supports Custom Alias:**
    - **Forbidden:** Select if a custom alias is not required.

      Note: If this is set to **Forbidden**, the **Alias** field will not display on the Add to Certificate Store page unless *Overwrite* is checked on the page.

    - **Optional:** Select if the custom alias is optional.
    - **Required:** Select if the custom alias is required.
  - **Private Key Handling:**
    - **Forbidden:** Select if a private key is not required; generally, applies to trust stores (e.g. Root CA certificates).
    - **Optional:** Select if the private key is optional; applies to store types that could represent either a Trust Store or End-Entity Store.
    - **Required:** Select if the private key is required; applies to stores that hold an End-Entity Certificate (server or client authorization).
  - **PFX Password Style:**
    - **Default:** Opt to have Keyfactor Command randomly generate a password.
    - **Custom:** Opt to allow a password to be entered and authenticated when enrolling a certificate through the Keyfactor Command Management Portal when installing a store of this type. The Custom option can be selected only if **Allow Custom Password** in the Application Settings is equal to *True*. For more details, see Application Settings: Enrollment Tab on page 609.

Custom Fields Tab

Custom fields define unique properties for the given certificate store type. Click **Add** on this tab to open the Add Custom Field dialog box.

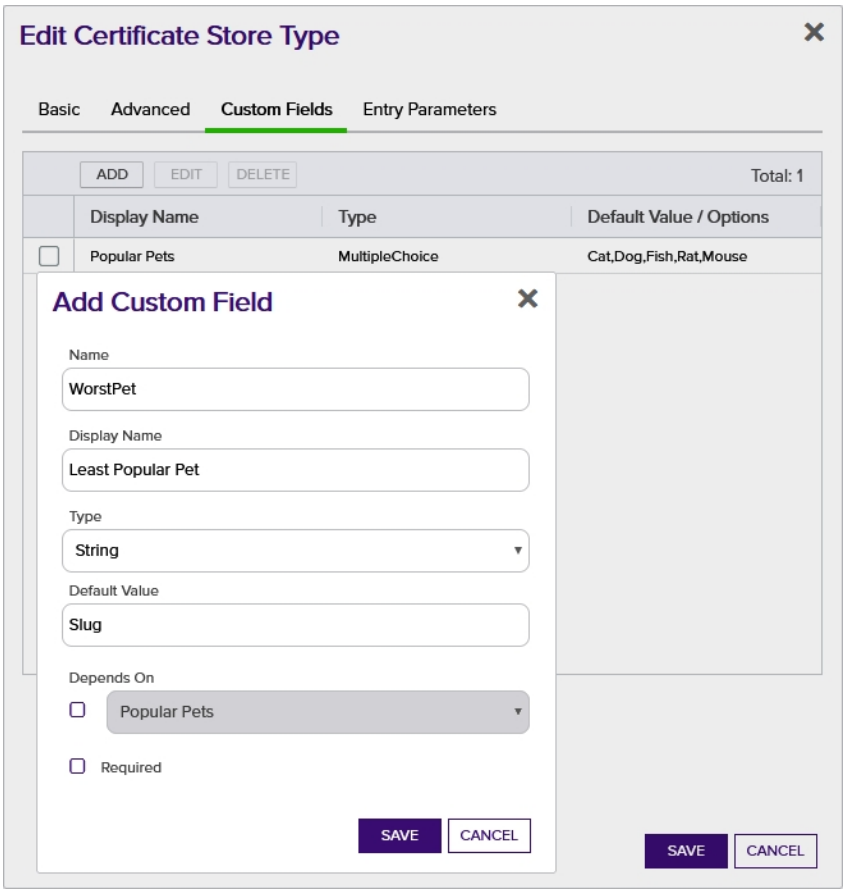


Figure 364: Add New Certificate Store Type: Custom Fields Tab

- **Name:** Enter the name submitted to the orchestrator and referenced in the extension module custom code.
- **Display Name:** Enter a user-friendly recognizable name.
- **Type:** Select whether parameter information is stored as a string, Boolean, multiple choice or secret.

  Note: This field cannot be modified on an edit.

- **Default Value / Multiple Choice Options:** Add a default value that will pre-populate the parameter field in the *Add New Certificate Store* dialog box. If you select a type of Multiple Choice, populate this field with a comma-separated list of multiple choice options for this parameter. If you select a type of Boolean, you will be given the option of True or False here.

- **Depends On:** Check this box if you have another custom field for this certificate store type and want to create a relationship between that one and this one. Then select the custom field on which this custom field depends in the dropdown. This option configures one custom field to display in the certificate store configuration dialog only if another custom field contains a value.
- **Required:** Select whether a value for this parameter must be entered before a certificate store can be added to Keyfactor Command.

Entry Parameters Tab

Entry parameters define unique properties that are required when performing management jobs on a certificate store of this type. Click **Add** on this tab to open the Add Entry Parameter dialog box.

Tip: What's the difference between custom fields and entry parameters?

- Custom fields are about the certificate store definition itself and are static. For example, you might use a custom field to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for custom fields are entered in the certificate store record when creating or editing the certificate store record.
- Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).

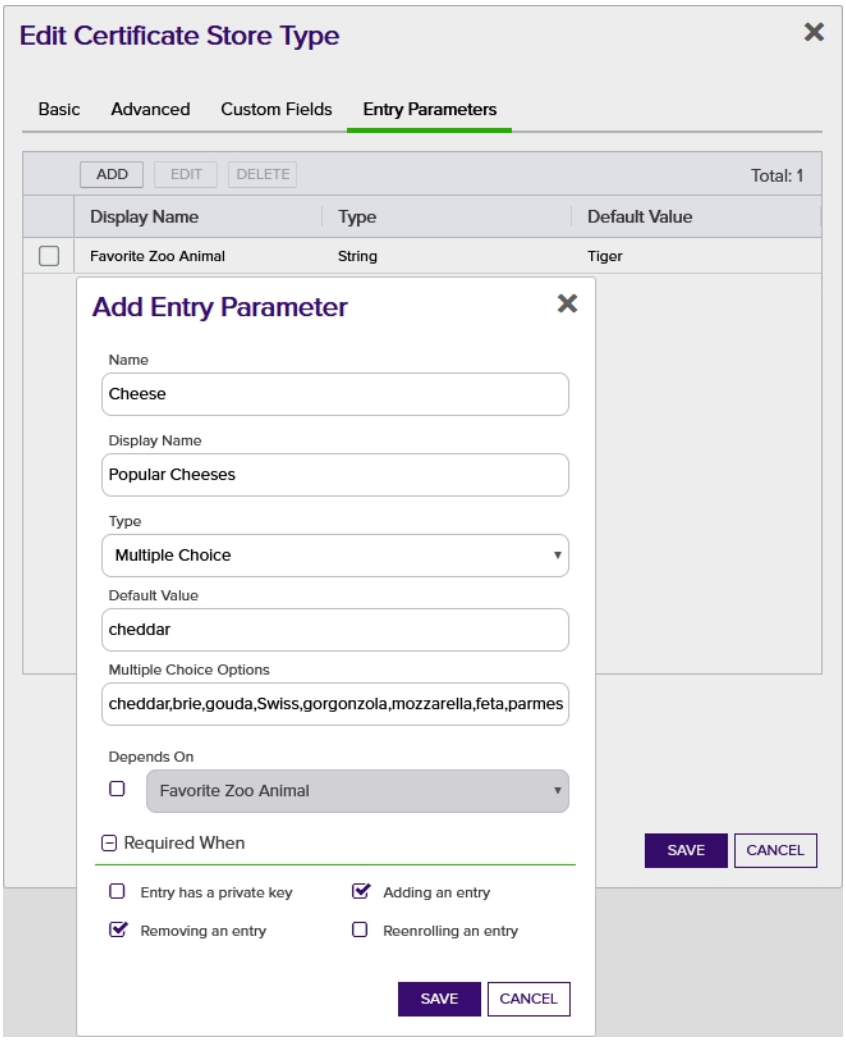


Figure 365: Add New Certificate Store Type: Entry Parameters Tab

- **Name:** Enter the name for the entry parameter. This value must be unique.
- **Display Name:** Enter a user-friendly recognizable name. This value must be unique.
- **Type:** Select whether parameter information is stored as a string, Boolean, multiple choice or secret.

> **Note:** This field cannot be modified on an edit.

- **Default Value:** Add a default value that will pre-populate the parameter field in the *Add New Certificate Store* dialog box. If you select a type of Boolean, you will be given the option of True, False, or Not Set here.
- **Multiple Choice Options:** Populate this field with a comma-separated list of multiple choice options if you selected a *Type* of multiple choice. This field will be grayed out if you selected a *Type* other than multiple choice.
- **Depends On:** Check this box if you have another entry parameter for this certificate store type and want to create a relationship between that one and this one. Then select the entry parameter on which this entry parameter depends in the dropdown. This option configures one entry parameter to display in the certificate store configuration dialog only if another entry parameter contains a value.
- **Required When:**
    - **Entry has a private key**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store.
    - **Adding an entry**: If set to *true*, a value must be provided for this field when configuring an add certificate job.
    - **Removing an entry**: If set to *true*, a value must be provided for this field when configuring a remove certificate job.
    - **Reenrolling an entry**: If set to *true*, a value must be provided for this field when configuring a reenrollment job.

4. Click **Save** to save the new certificate store type.

Note: Built-in certificate store types cannot be edited.

Deleting a Certificate Store Type

You may delete one store type at a time.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Certificate Stores > Modify
Certificate Stores > Read

To delete a certificate store type:

1. In the Management Portal, browse to *System Settings Icon* > *Certificate Store Types*.
2. On the Certificate Store Types page, highlight the row in the grid of the certificate store type to delete and click **Delete** at the top of the grid or right-click the type in the grid and choose

Delete from the right-click menu.

3. On the Confirm Operation alert, click **OK** to confirm or **Cancel** to cancel the operation.

2.1.11.4 Certificate Metadata

Using user-defined certificate metadata you can tag certificates with additional information you want to assign to certificates at the point of enrollment, such as points of contact or certificate/app owners. Metadata fields can be defined as being *required* or *optional* during enrollment. The data from the metadata fields can then be used for queries and alerts in the Management Portal.

First, you must add all the metadata fields you will use across the platform via *System Settings Icon* ⚙ *> Certificate Metadata* (see Metadata Field Operations below). These *system-wide* settings will then become the default metadata settings for all templates and they will be assigned to certificates during enrollment via the selected template. You may choose to modify the *system-wide* metadata field(s) for specific templates by creating *template-specific* metadata settings. See Certificate Template Operations on page 381 and Enrollment on page 135 for more information.

Tip: Click the help icon (❓) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Metadata Field Operations

To select a single row in the certificate metadata field grid, click to highlight it and then select an operation from either the top of the grid or the right-click menu.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Metadata > Types > Read
Metadata > Types > Modify

Adding or Modifying a Metadata Field

To create a new metadata field or edit an existing one:

1. In the Management Portal, browse to the *System Settings Icon* ⚙ *> Certificate Metadata.*
2. On the Certificate Metadata page, click **Add** to create a new metadata field, or, to edit an existing one, double-click the row in the metadata grid, right-click the row and choose **Edit** from the right-click menu, or highlight the row in the grid and click **Edit** at the top of the grid.

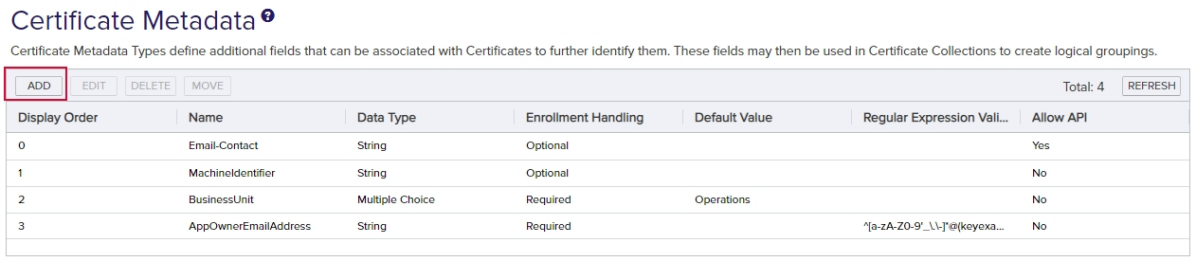


Figure 366: Certificate Metadata

3. In the Metadata Edit dialog, enter a **Name** for your metadata field. This name appears in interfaces where you can use metadata, such as certificate details dialogs, alert dialogs, certificate imports and certificate requests. Once this field has a value associated with it for at least one certificate, you cannot change this name. The metadata name field cannot contain spaces; dashes and underscores are supported.

   Important: Be sure to review the list of existing queryable certificate fields on the Certificate Search Page on page 34 before adding a new metadata field, so you do not add a field of the same name or alias as an existing field. Doing so would cause a search or alert on that field to fail. For example, do not create a metadata field called *NetBIOSRequester* or its alias *RequesterName*, as this would match is an existing certificate field, and having a metadata field with this name would create issues.

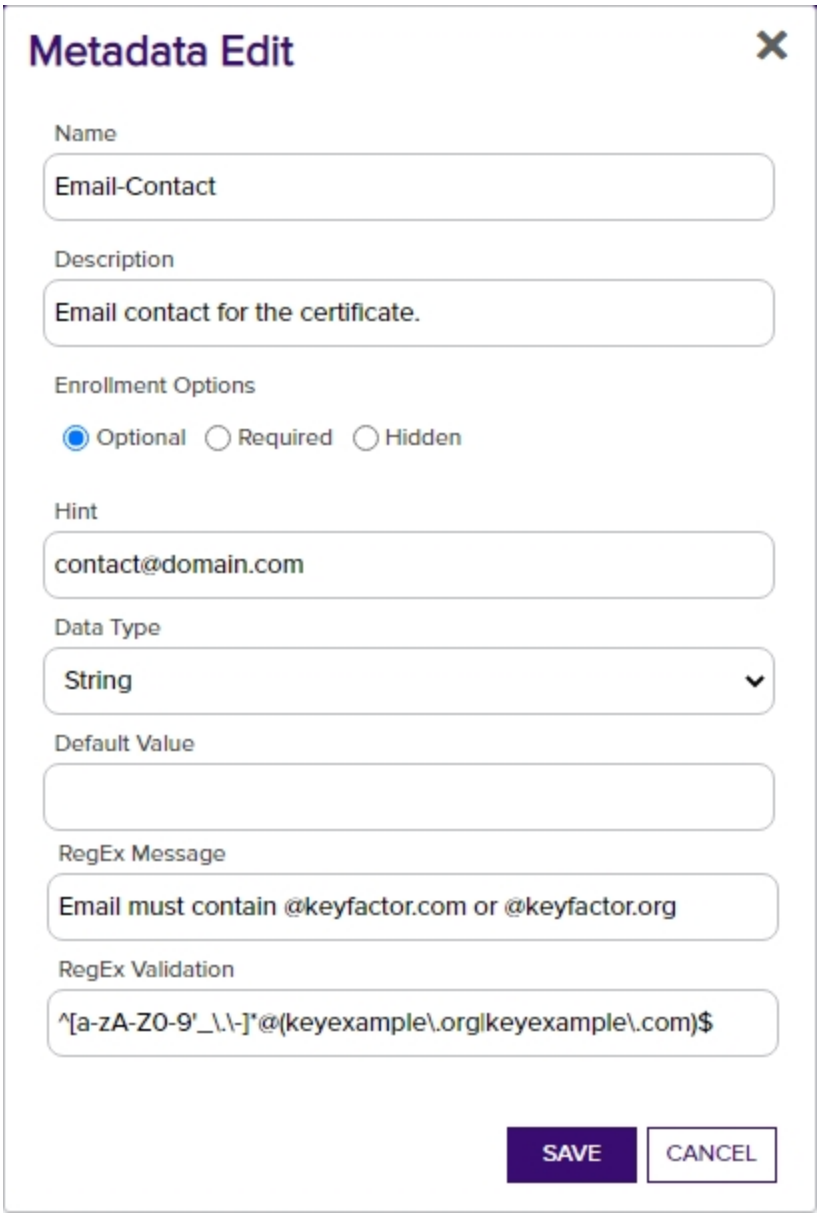


Figure 367: Create or Edit Certificate Metadata Field

4. Enter a **Description** for the metadata field.
5. The **Enrollment Options** provide three possible settings for the metadata field:
    - Select the **Optional** radio button to allow users the option to either enter a value or not enter a value in the field when populating metadata fields.

- Select the **Required** radio button to force users to enter a value in the field when populating metadata fields. Required fields will be marked with *Required next to the field label on the Certificate Details dialog for a certificate and on the certificate enrollment pages.
- To hide the field on the enrollment pages (see Enrollment on page 135), select the **Hidden** radio button. Selecting the **Hidden** option does not hide the field in the certificate details (see Metadata Tab on page 20) or on the Add Certificate page (see Add Certificate on page 74).

6. Enter a short hint in the **Hint** field. This hint appears in unpopulated metadata string, integer, big text and date fields on editing interfaces to provide the user with a clue as to what type of data should be entered in the field.

   Note: The Hint field is not used for some selections of the Data Type field (see the next step) and will disappear from the screen if a Data Type that does not use a Hint is selected.

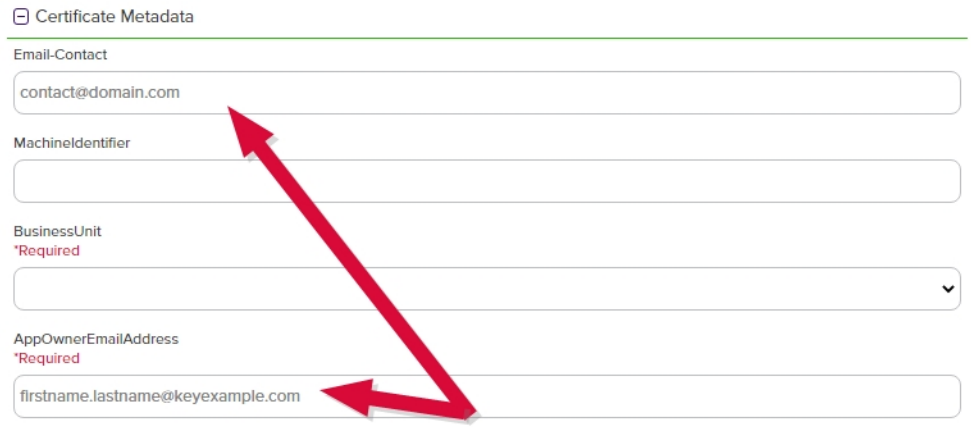


Figure 368: Metadata Hints in a Certificate Details Dialog

7. Select the **Data Type** for the field in the dropdown. The available field types are String (alphanumeric), Integer (whole numbers), Date, Multiple Choice, Big Text, and Boolean (True/False). String fields are limited to 400 characters. Big text fields are limited to 4000 characters. String fields support additional indexing, and so may be preferable for large databases where possible. The data type cannot be edited if the metadata field is associated with any certificate values.

   The remaining fields on the dialog—plus the *Hint*—will vary depending on the data type selected. Table 72: Certificate Metadata Data Type Dialog Options shows the fields that appear based on the data type selected.

Table 72: Certificate Metadata Data Type Dialog Options

Data Type	Character Limit	Hint	Default Value	RegEx Message	RegEx Validation	Options
String	400 alpha-numeric with indexing	✓	✓	✓	✓	
Integer		✓	✓			
Date		✓				
Boolean			✓			
Multiple Choice	4,000		✓			✓
Big Text	4000	✓				

8. To set a default value with which to pre-populate the metadata field for new certificate requests made using the Management Portal enrollment pages, enter the desired value in the **Default Value** box, or, for Boolean fields, select the desired radio button. The default value option appears for string, integer, Boolean and multiple choice fields.

9. For string fields, you can choose to enter a regular expression against which entered data will be validated in the **RegEx Validation** field. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the **RegEx Message** field. The example regular expression shown in Figure 367: Create or Edit Certificate Metadata Field is:

   ```
   ^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$
   ```

   This regular expression specifies that the data entered in the field must consist of some number of characters prior to the *@* made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either *@keyexample.org* or *keyexample.com*. For more examples of regular expressions, see Regular Expressions on page 402.

10. For multiple choice fields, enter the series of values that should appear in the field dropdown as a comma delimited list in the **Options** field.

    For example:

Accounting,HR,IT,Marketing,Sales

Note: The multiple choice options are displayed in the order entered in the comma delimited list. When a user selects a multiple choice value in a metadata field while editing a certificate, the value is saved to the database as the string (e.g. Marketing). Subsequently editing the series of values for the metadata field or rearranging them will not affect existing certificates configured with values for this field.

11. Click **Save** to save your metadata field.

Sorting Metadata Fields

You may change the display order for metadata fields. This affects how the fields display on the certificate details, certificate template details when configuring the metadata tab, and on enrollment pages.

To change the display order of a metadata field:

1. Browse to *System Settings Icon* > *Certificate Metadata.*
2. Right-click a grid row and choose **Move** from the right-click menu, or highlight the row in the grid and click **Move** at the top of the grid.
3. In the Display Order dialog enter the desired display order number and click **Save**. The value entered must fall without the current display order range. For example, if the current range is 0-12, enter 12 to move a field to the end of the list, not 13. The metadata field will move to the entered display order row and the metadata fields from the rows above and below will be re-ordered.

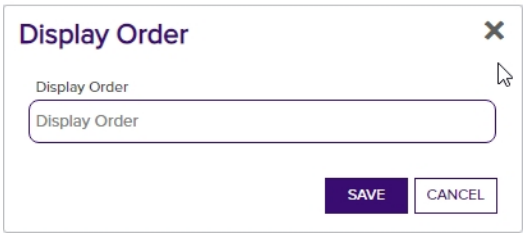


Figure 369: Metadata Display Order

Deleting a Metadata Field

Metadata fields cannot be deleted if they are associated with any certificate values.

To delete a metadata field:

1. Browse to *System Settings Icon* > *Certificate Metadata.*
2. Right-click a grid row and choose **Delete** from the right-click menu, or highlight the row in the grid and click **Delete** at the top of the grid.

2.1.11.5 Audit Log

PKI is more than Keyfactor Command, CAs, and certificates. It also includes the people and polices that interact with these entities. It is therefore critical to track the actions taken within Keyfactor Command that enable management of all entities that make up a PKI, as most attack vectors are only exposed internally. The Keyfactor Command audit logs are an immutable record of all changes made to the state of the application.

The information collected in the audit logs is available for viewing and analysis by several means:

- The data is available for viewing within the Keyfactor Command Management Portal, where a search tool may be used to search for specific logs (see Using the Audit Log Search Feature on the next page).
- The data is output to text-based logs on the Keyfactor Command server and stored for 14 days, by default (see Log Monitoring on page 794). From here, the logs may be collected by a centralized logging solution for analysis.
- The data is output to the Windows event log on the Keyfactor Command server in the Windows application event log. From here, the logs may be collected by a centralized logging solution for analysis. See Keyfactor Command Windows Event IDs on page 806. When analyzing audit logs as written to the Windows event log, it can be helpful to have the translations for the operation codes handy (see Audit Log Reference Codes on page 727). Audit log failures (when Keyfactor Command fails to log to the audit log) are also logged to the Windows event log.
- The data may optionally be copied in real time to a separate server for analysis with a centralized logging solution (e.g. rsyslog, Logstash). For more information, see Audit Log Output to a Centralized Logging Solution on page 805.

Any activity that triggers an audit flag generates an audit record. Auditable activities include actions (e.g. creation, change, deletion) on records in Keyfactor Command that have been configured as auditable (e.g. Certificates, Security, Templates, Application Settings). For a complete list of Keyfactor Command activity that is tracked through the audit log, see Audit Log Reference Codes on page 727.

The audit log page in the Keyfactor Command Management Portal allows you to view all the audit logs stored in Keyfactor Command and perform searches on them. Audit logs are stored for seven years, by default (see Application Settings: Auditing Tab on page 608).

The audit log grid includes these fields:

- Level
  The logging level of the message. Most messages are generated at Information level.
- Category
  The area of Keyfactor Command that generated the audit log (see Audit Log Categories on page 729).
- Message
  The audit log message. The message is made up of the user who took the auditable action, the action the user took, the category the user acted upon, and the name of the object acted upon.
- Timestamp
  The time and date that the message was generated.

The grid can be sorted by clicking on a column header. All columns except Message may be sorted. Click the column header again to reverse the sort order. The grid columns can be arranged in any order desired by click-holding and dragging the header of the column you wish to move. The column widths may be adjusted by click-holding and dragging the line separating two column headers.

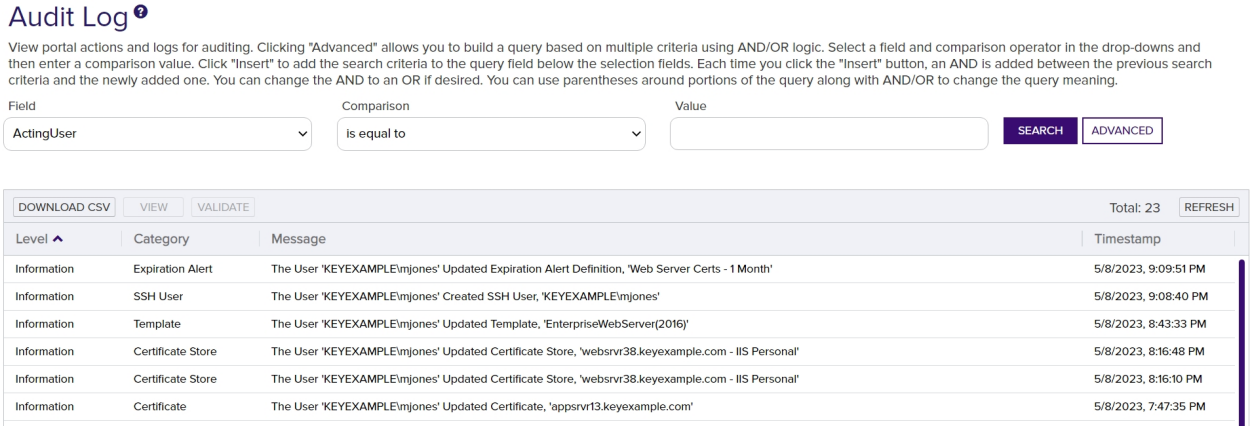


Figure 370: Audit Log

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Using the Audit Log Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

ActingUser	Name

The user who performed the audited action, in the format DOMAIN\username for Active Directory authentication or typically just the username for an identity provider other than Active Directory. For actions initiated by the Keyfactor Command Service, this will be *Timer Service*. Supports the %ME% token (see Advanced Searches on page 720).

Level

The logging level of the message:

- Information
  A successful operation that changes the state of the data in the application
- Warning
  Notification of a possible malicious access attempt (e.g. an unauthorized user attempting to access a web page)
- Failure
  Notification that a user was denied access to an activity (this can be used to alert to a possible internal role security issue)

Timestamp

The time at which an action took place. Supports the %TODAY% token (see Advanced Searches on page 720).

Category

The area of the product in which the auditable activity occurred. This list is built dynamically to show only those categories that are actually in your audit log. Select a category (e.g. Template) and for most

The name of the object being audited. The name of the object is related to the category of auditable activity. If the category is template, the name will be the template name. If the category is SSH user, the name will be the username of the user owning the SSH key. If the category is expiration alert, the name will be the expiration alert name. Some category to name relationships are more clear than others.

For example, in the following audit message for a certificate enrollment, the name is the DN of the certificate:

```
The user 'KEYEXAMPLE\ggant' Created Certificate,
'CN=appsrvr12.keyexample.com,L=Chicago,ST=IL,C=US'
```

In the following workflow instance message, the name is the entire title of the workflow instance:

```
The User 'KEYEXAMPLE\mjones' Completed Workflow
Instance, 'KEYEXAMPLE\mjones is enrolling for a
certificate with CN=websrvr12.keyexample.com.'
```

In the following certificate collection message, the name is the name of the certificate collection that was created:

```
The user 'KEYEXAMPLE\jsmith' Created Certificate
Query, 'Revoked Certs'
```

When you open the details for an audit log record, the name appears at the top of the details dialog as the second part of the dialog title (see View: Audit Log Details on page 723).

Operation

The type of operation performed. See Audit Log Operations on page 727 for a complete list of the available operations.

categories an optional subsearch field (e.g. Template Defaults for templates) to find entries related to that category and optional subsearch field (e.g. any changes made to template default settings). See Audit Log Categories on page 729 for a complete list of possible categories.

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.

- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Tip: The *Message* field in the audit log grid is built from the *Acting User*, *Operation*, *Category*, and *Name* fields and is not searchable as the Message. Instead, search by Acting User, Operation, Category, and/or Name.

When you select Category in the query field, a fourth dropdown will appear. This *Property Field* allows you to further refine the search. The options available in this field vary depending on the selection made in the comparison value. Select *Any* to display all of the results for the selected category search combination. Select a specific value in the property field to display all the audit records that had changes to the selected field.

Example: To see only changes made to the template default settings for certificate templates, select *Category* in the query field, *is equal to* in the comparison operator, *Template* in the comparison value, *Template Defaults* in the property field, and click **Search**.

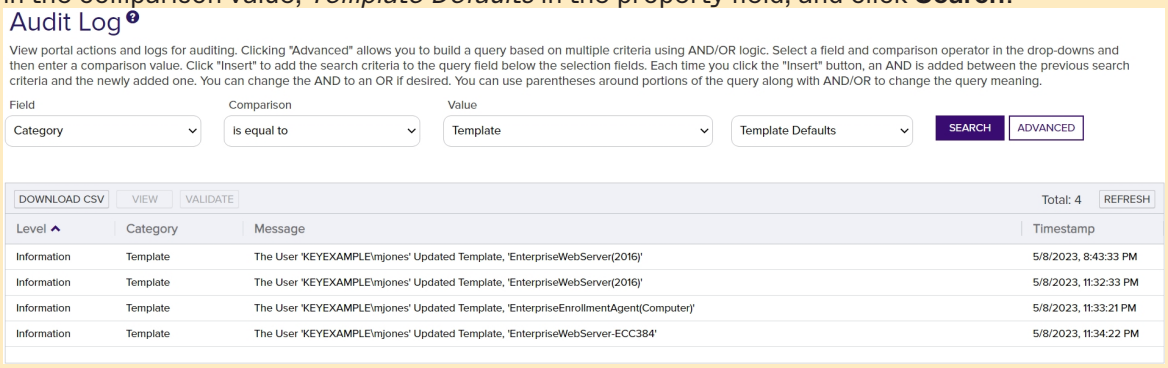


Figure 371: Audit Log Search Selections for Template Property Field Search

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

In addition to the options available in the query builder, three special values can be used in selected searches by typing them in directly:

- %TODAY%
  Use the TODAY special value in place of a specific date in date queries. This option supports math operations, so you can use TODAY-10 or TODAY+30. The built-in *Certificates Expiring in 7 Days* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME%
  Use the ME special value in place of a specific domain\user name in queries that match a domain\user name. The built-in *My Certificates* collection uses this special value (see Certificate Collection Manager on page 85).
- %ME-AN%
  Use the ME-AN special value in place of a specific user name excluding the domain. This is beneficial in environments with multiple domains where there is a desire to query for a user's certificates even if they were requested across multiple domains.

Important: The special query options of %TODAY%, %ME%, and %ME-AN% are only supported in uppercase. Lowercase equivalents (e.g. %me%) cannot be substituted.

Audit Log Operations

The audit log page in the Keyfactor Command Management Portal allows you to perform searches for all the audit logs stored in Keyfactor Command, view details for them, validate that they have not been tampered with, and output selections of them in CSV format.

Download CSV

Click the **Download CSV** button at the top of the audit log grid to generate and download a comma-delimited CSV file containing all audit log records per the search criteria applied to the grid. The CSV file will contain the information shown in Table 73: Audit Download CSV Records for each exported record.

Table 73: Audit Download CSV Records

Field	Description
Id	Sequential Internal reference number

Field	Description
Timestamp	The date and time the auditable change was made.
Message	The message displayed on the audit log grid. This field contains a human-readable summary of the change and is made up of the user who took the auditable action, the action the user took, the category the user acted upon, and the name of the object acted upon.
Operation	The operation type (e.g. Created, Updated, Deleted). For a list of possible operations, see Audit Log Operations on page 727.
Level	The logging level of the message (e.g. Info, Warning). Most messages are generated at Information level.
User	The user taking the action that generated that audit log, generally in DOMAIN\user-name format, though for actions initiated by the Keyfactor Command Service, this will be *Timer Service*.
Category	The area of the product in which the change was made (e.g. Certificates, Templates, Application Settings) as per the available values in the *category* field in the audit grid. For a list of possible categories, see Audit Log Categories on page 729.
Name	The specific object the action was taken on (e.g. the template name for a template change or the application setting name for an application setting change).

XMLMessage	The details of the change that was made in XML format. This field contains both the before state and the after state where applicable (e.g. an application setting that was configured as *true* before the change and *false* after the change). For example, this entry indicates that a change was made to the key retention policy (the template name the change was made to is specified in the Name field) to change the number of days for retention from four days to seven days:
	
	`<AuditAction>`
	`  <ModelState>`
	`  <Template>`
	`  <KeyRetention enum-type=CSS.CMS.Core.Enums.KeyRetentionPolicy">3</KeyRetention>`
	`  <KeyRetentionDays>7</KeyRetentionDays>`
	`  <AllowedEnrollmentTypesDisplay ienumerable="true">`
	`  <string>PFX Enrollment</string>`
	`  <string>CSR Enrollment</string>`
	`  <string>CSR Generation</string>`
	`  </AllowedEnrollmentTypesDisplay>`
	`  </Template>`
	`  </ModelState>`

Field	Description

	`<PreviousModelState>`
	`<Template>`
	`<KeyRetention enum-type="CSS.CMS.Core.Enums.KeyRetentionPolicy">3</KeyRetention>`
	`<KeyRetentionDays>4</KeyRetentionDays>`
	`<AllowedEnrollmentTypesDisplay ienumerable="true">`
	`<string>PFX Enrollment</string>`
	`<string>CSR Enrollment</string>`
	`<string>CSR Generation</string>`
	`</AllowedEnrollmentTypesDisplay>`
	`</Template>`
	`</PreviousModelState>`
	`</AuditAction>"`

View: Audit Log Details

To view audit log details for an audit log record, double-click the audit log entry in the audit log grid, right-click the row in the grid and choose **View** from the right-click menu, or highlight the row in the grid and click **View** at the top of the grid. The information on the detail dialog will vary depending on the type of activity that was logged.

The contents of the audit log details dialog will vary depending on the category and object type audited and whether the log item is a new entry or has been updated. The details dialog has four sections.

Name

The Keyfactor Command audit **Name** for the selected audit log entry is in the gray title bar at the top of the dialog. This is a useful field to use in the search criteria.

Entry Metadata

Directly below the **Name** at the top left of the dialog is the **Entry Metadata** section, which displays the internal metadata information about the currently displayed detail record:

- Operation
  The type of activity that generated the audit log record (see Audit Log Operations on page 721).
- Time
  The time and date that the audit log entry was generated.
- User
  The user who carried out the activity that generated the audit log.
- Category
  The area of the product in which the auditable activity occurred (see Audit Log Categories on page 729).

- Validation Status
  Whether the audit log entry in the database is valid or invalid.

Selecting a different entry in the **Related Entries** section will change the display in this section.

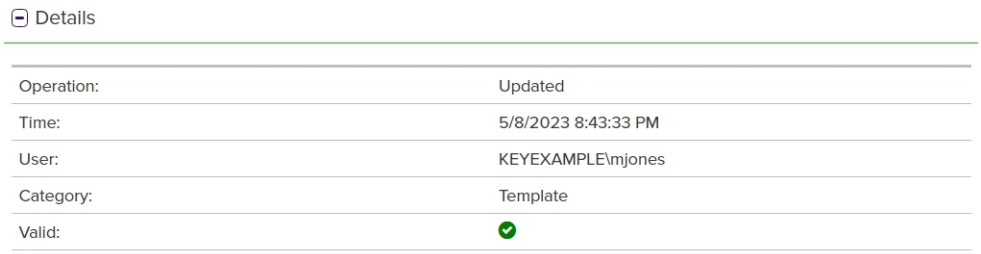


Figure 372: Audit Log Details: Entry Metadata Section

Related Entries

The **Related Entries** tab displays the history of all the related audit log items (e.g. changes to the same template or certificate) for the selected audit log entry. Click a row in the related entries grid and click **View** to update the details dialog with the details of the audit log item for the selected related entry.

The related entries can be sorted by clicking on a the *Time* or *User* column headers in the results grid. Click the column header again to reverse the sort order.

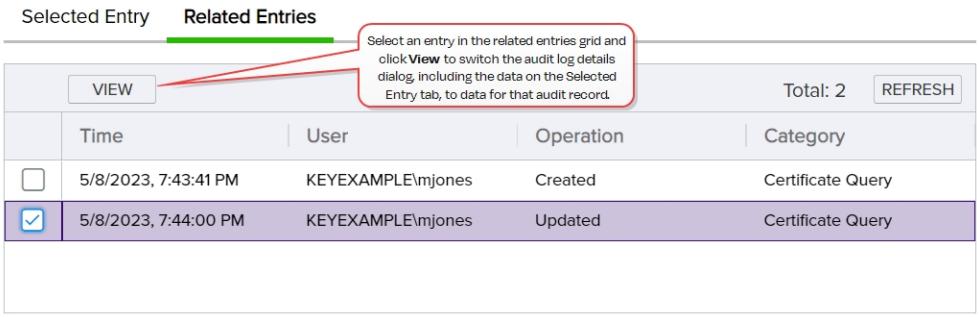


Figure 373: Audit Log Details: Related Entries Section

Note: The Related Entries tab includes all entries, including the initial entry that you opened to reach the tab.

Selected Entries: Audit Details Pane

The Selected Entry tab of the audit log details dialog will either have one column (for new, or single event, entries) or two (for updated items) showing the details of the auditable action.

The title of a single column pane changes depending on the audit entry event that triggered the entry. It is made up of the category and operation performed to create the entry. The details displayed vary depending on the type object being audited.

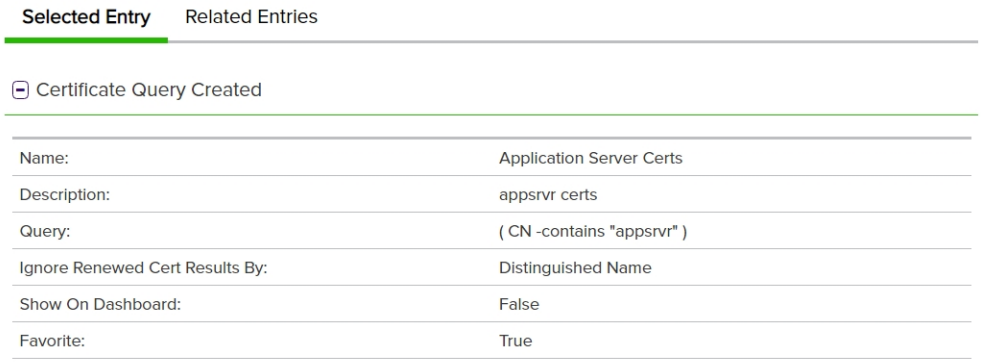


Figure 374: Audit Log Details: Single Column Audit Details Pane

The two column pane includes **Before Changes** and **After Changes** sections. Only those details that have a different value as a result of a particular audit event will be displayed. Changed fields with sensitive data will display as ******.

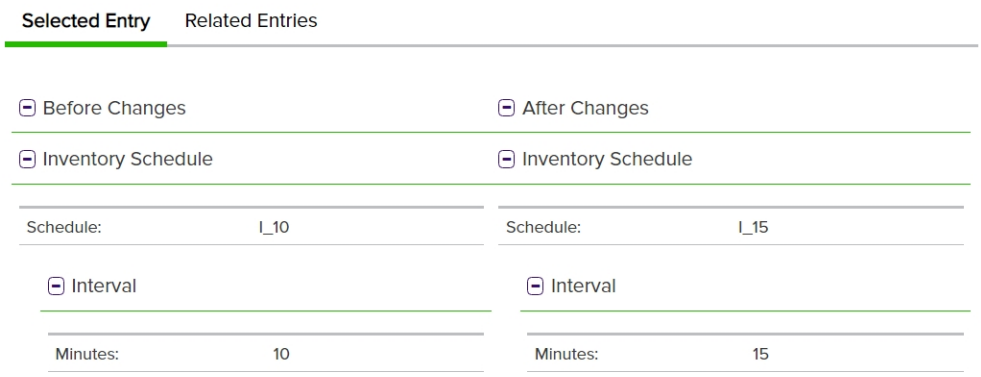


Figure 375: Audit Log Details: Two Column Audit Details Pane

Note: Updates where a field had no value before the update will appear in the single column format.

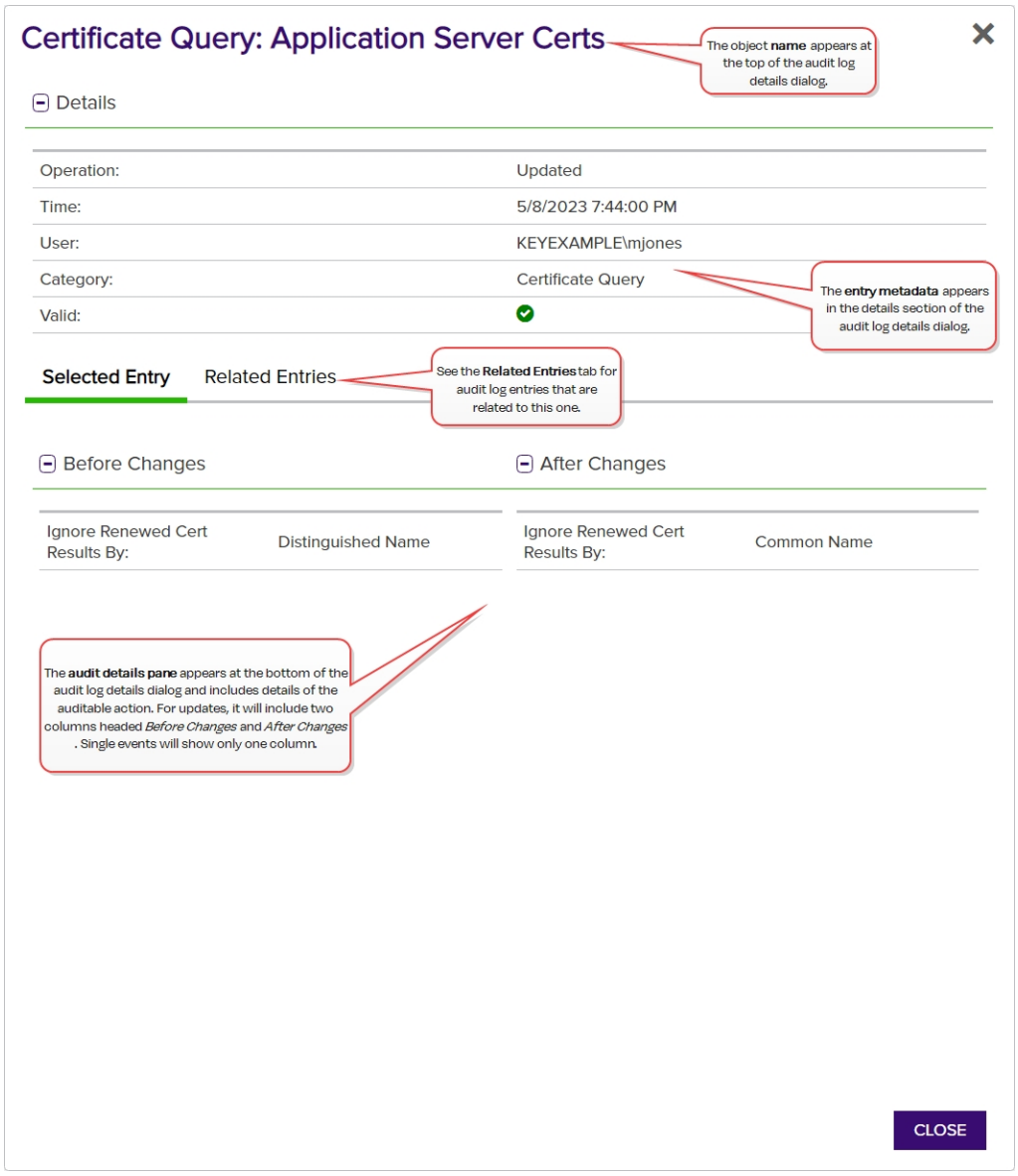


Figure 376: Audit Log Details Dialog

Click **Close** to close the details dialog.

Validate

Highlight a row in the audit log grid and click the **Validate** button to verify whether the selected item is valid or not valid. This function checks the integrity of the audit log data for that grid row to determine whether the data has been tampered with. If the status of the selected item is valid, the validate dialog will indicate this. If the selected item has been tampered with, the validate dialog will indicate that the selected item is not valid.

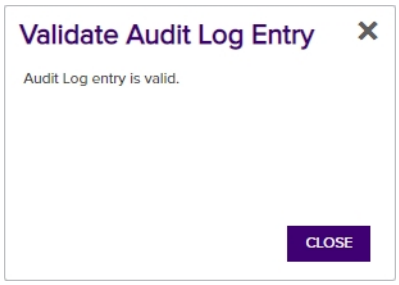


Figure 377: Audit Log Record is Valid

The validation status of any audit log item can also be viewed in the details dialog, where a status of **Valid:** ✔ or **Valid:** ✖ will be shown.

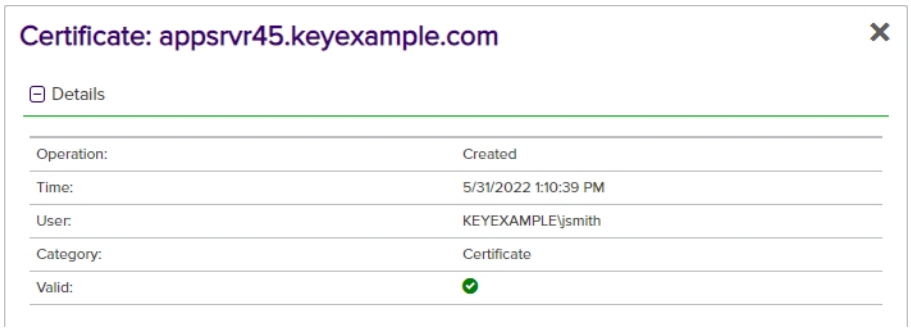


Figure 378: Audit Log Details Showing Valid Status

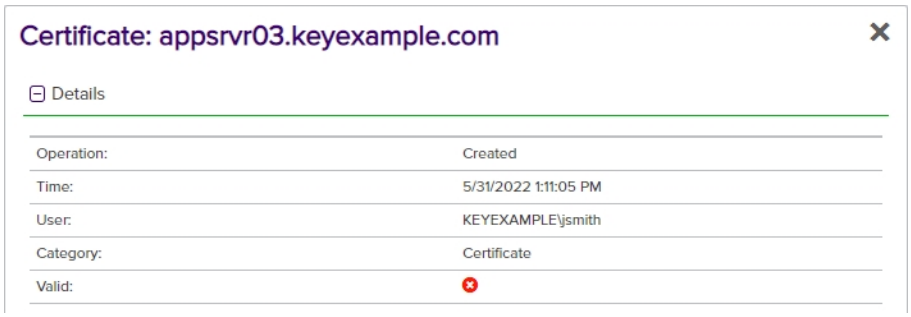


Figure 379: Audit Log Details Showing Invalid Status

Audit Log Reference Codes

The Keyfactor Command audit logs are a record of historical changes that have been made within the product to key systems. The following shows the full list of currently audited areas (areas of the product) and operations (types of activity). The equivalent numeric codes are included for those interested in viewing or analyzing raw log data.

Audit Log Operations

The type of operation performed.

Table 74: Audit Operations

Value	Description
1	Created
2	Updated
3	Deleted
4	Approved
5	Denied
6	Revoked
7	Downloaded
8	Deleted Private Key
9	Renewed
10	Encountered
11	Scheduled Replacement
12	Recovered
13	Imported
14	Removed from Hold
15	Scheduled Add
16	Scheduled Removal
17	Download with Private Key
18	Scheduled
19	Reset
20	Disapproved
21	Restarted
22	Sent

Value	Description
23	Failed
24	Completed
25	Rejected

Audit Log Categories

The area of the product in which the auditable activity occurred. The subcategory name is primarily used in the Keyfactor API or when reviewing downloaded CSV files.

Table 75: Audit Categories

Value	**Subcategory Name**	Description
2001	Certificate	Certificate
2001	AuditingCertificateScheduledReplacement	Auditing Certificate Scheduled Replacement
2001	AuditingCertificateRequest	Certificate Request
2002	ApiApplication	API Application
2003	Template	Template
2004	CertificateQuery	Certificate Collection/Query
2005	ExpirationAlert	Expiration Alert
2005	ExpirationAlertDefinitionContextModel	Expiration Alert
2006	PendingAlert	Pending Alert
2006	PendingAlertDefinitionContextModel	Pending Alert
2007	ApplicationSetting	Application Setting
2008	IssuedAlert	Issued Alert
2008	IssuedAlertDefinitionContextModel	Issued Alert
2009	DeniedAlert	Denied Alert
2009	DeniedAlertDefinitionContextModel	Denied Alert

Value	**Subcategory Name**	Description
2010	ADIdentityModel	Security Identity
2011	SecurityRole	Security Role
2012	AuthorizationFailure	Authorization Failure
2013	CertificateSigningRequest	CSR
2014	ServerGroup	SSH Server Group
2015	Server	SSH Server
2016	DiscoveredKey	Rogue Key for Logon
2016	Key	SSH Key
2017	ServiceAccount	SSH Service Account
2018	Logon	SSH Logon
2019	SshUser	SSH User
2020	KeyRotationAlertDefinitionContextModel	SSH Key Rotation Alert
2021	CertificateStore	Certificate Store
2022	JobType	Orchestrator Job Type
2023	AgentSchedule	Orchestrator Job
2024	BulkAgentSchedule	Bulk Orchestrator Job
2025	CertificateStoreContainer	Store Container
2026	Agent	Orchestrator
2027	RevocationMonitoring	Monitoring
2028	License	License
2029	WorkflowDefinition	Workflow Definition
2030	WorkflowInstance	Workflow Instance
2031	WorkflowInstanceSignal	Workflow Instance Signal

Value	Subcategory Name	Description
2032	IdentityProvider	Identity Provider
2033	RoleClaimDefinition	Claim Definition
2034	PermissionSet	Permission Set

Tip: The Category code of the auditable activity matches the Windows Event ID of the activity.

Audit Logging Specifics

While the Keyfactor Command audit log functionality covers the entire product, the following areas may be of particular interest.

Access Control

When a user tries to access a page in the Management Portal or an API endpoint that they don't have access to, they will receive an error and a warning will be logged in the audit log.

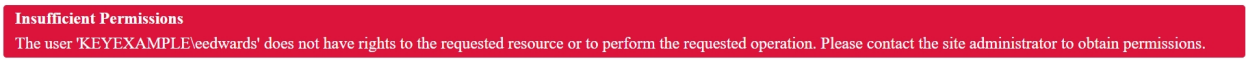


Figure 380: Management Portal Access Denied Message

The audit log shows the level as *Warning* and the category as *Authorization Failure* with a message detailing the user and the requested page.

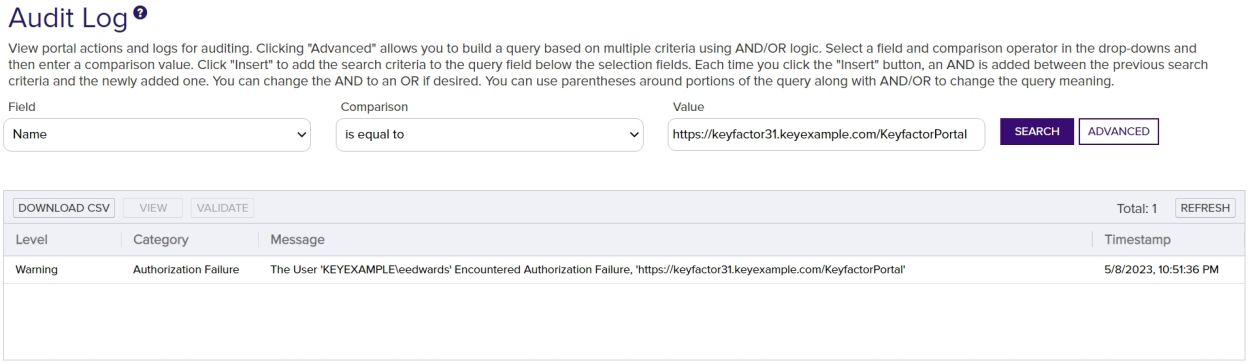


Figure 381: Audit Log Authorization Failure Messages

Click **View** to see the details dialog:

- Username
  The user making the page request.

- Request Route
  The page the user requested.
- Request Type
  Either *API Endpoint* or *Portal Page*.
- HTTP Verb
  This appears for both API requests and portal requests. For API requests, this can help to determine which action was denied.
- User's Roles
  The security role or roles that the user holds (see Security Roles and Claims on page 622). A role will not be listed if the user denied access is not a user in Keyfactor Command.

For more information about the audit log details, see View: Audit Log Details on page 723.

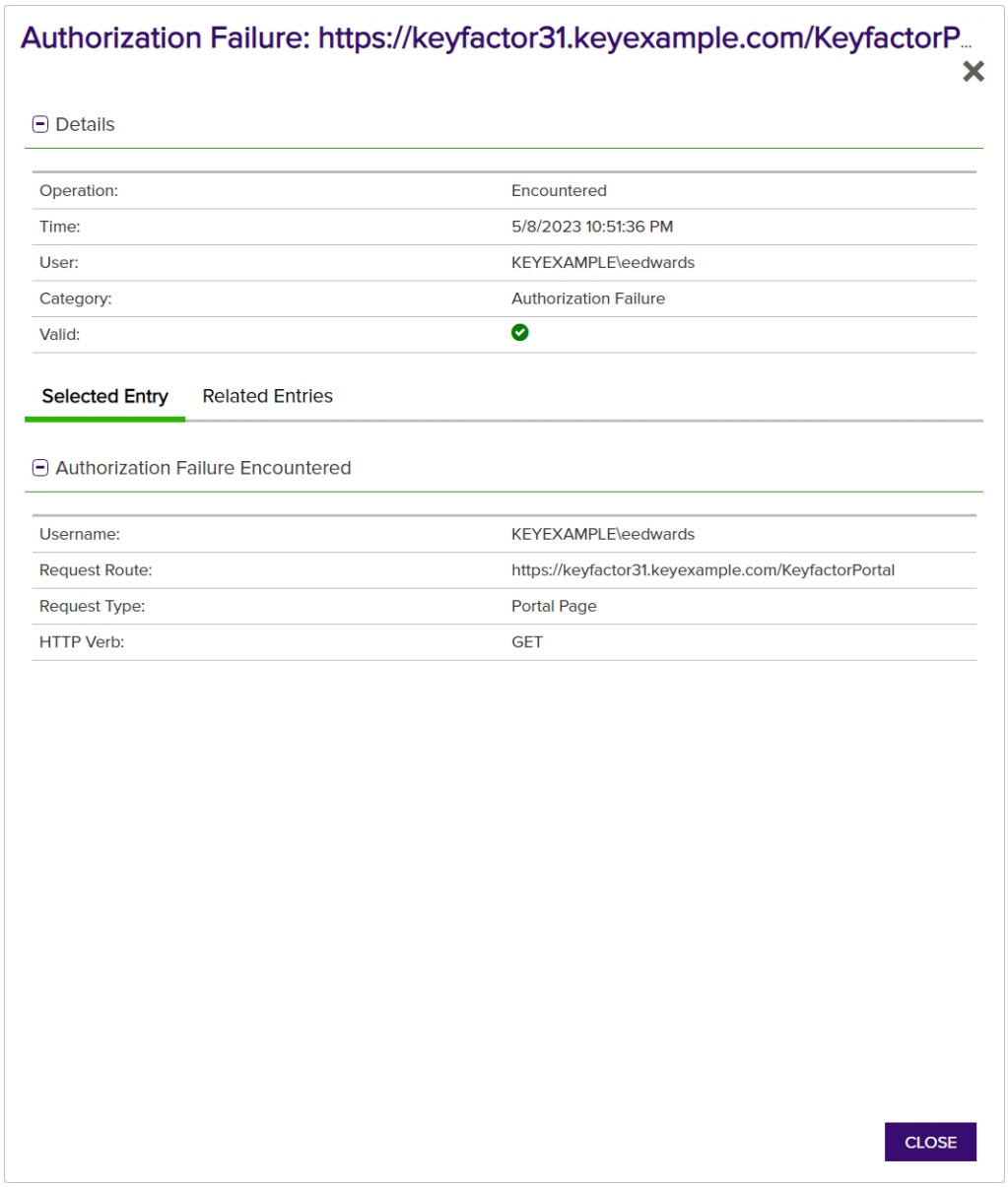


Figure 382: Authorization Failure Audit Log Detail

Certificate Operations

Tracking of operations related to certificates is especially extensive. Certificate-related operations that are audited include:

- Certificate revocation (Category: Certificate)
- Certificate download (Category: Certificate)
- Enrollment for certificates via PFX enrollment and CSR enrollment (Category: Certificate)
- Certificate renewal via one-click or seeded renewal (Category: Certificate)

- CSR generation, re-download and deletion (Category: CSR)
- Approval of certificate requests made using templates requiring manager approval at the CA level (Category: Certificate Request—see also Workflow on page 737)
- Certificate deletion (Category: Certificate)
- Certificate metadata operations—addition of or updates to metadata tags on certificates (Category: Certificate)
- Certificate collection creation or modification (Category: Certificate)
- Addition of certificates to and removal from certificate stores (Category: Certificate Store)

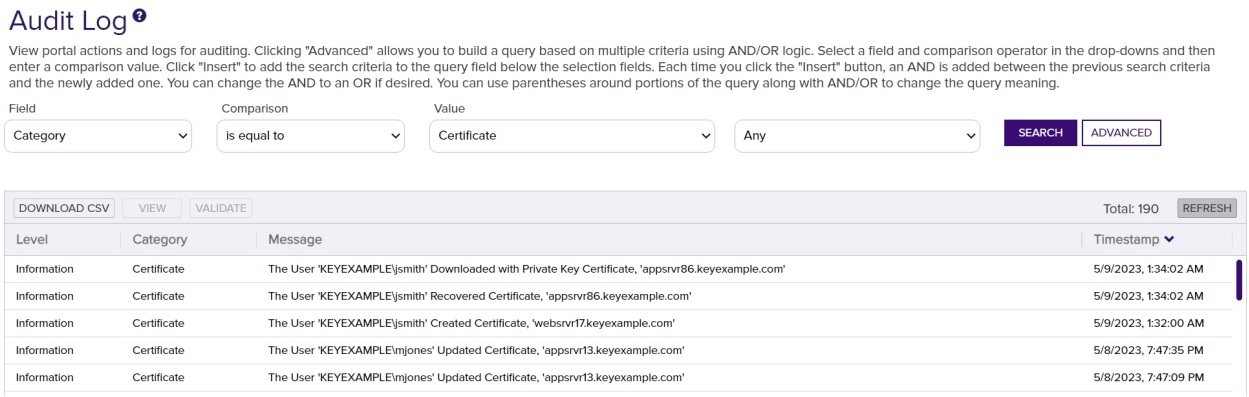


Figure 383: Audit Logs for Certificates

Security

The management of security identities and roles to limit access to the Keyfactor Command Management Portal and Keyfactor API generates audit log entries as roles and identities are created, updated, and deleted. This includes the granting of permissions to roles and the assigning of roles to identities (these are considered updates). The Security Identity and Security Role categories do not cover any attempts to access the system. These are tracked separately using the Authorization Failure category (see Access Control on page 731). Successful authorizations are not logged.

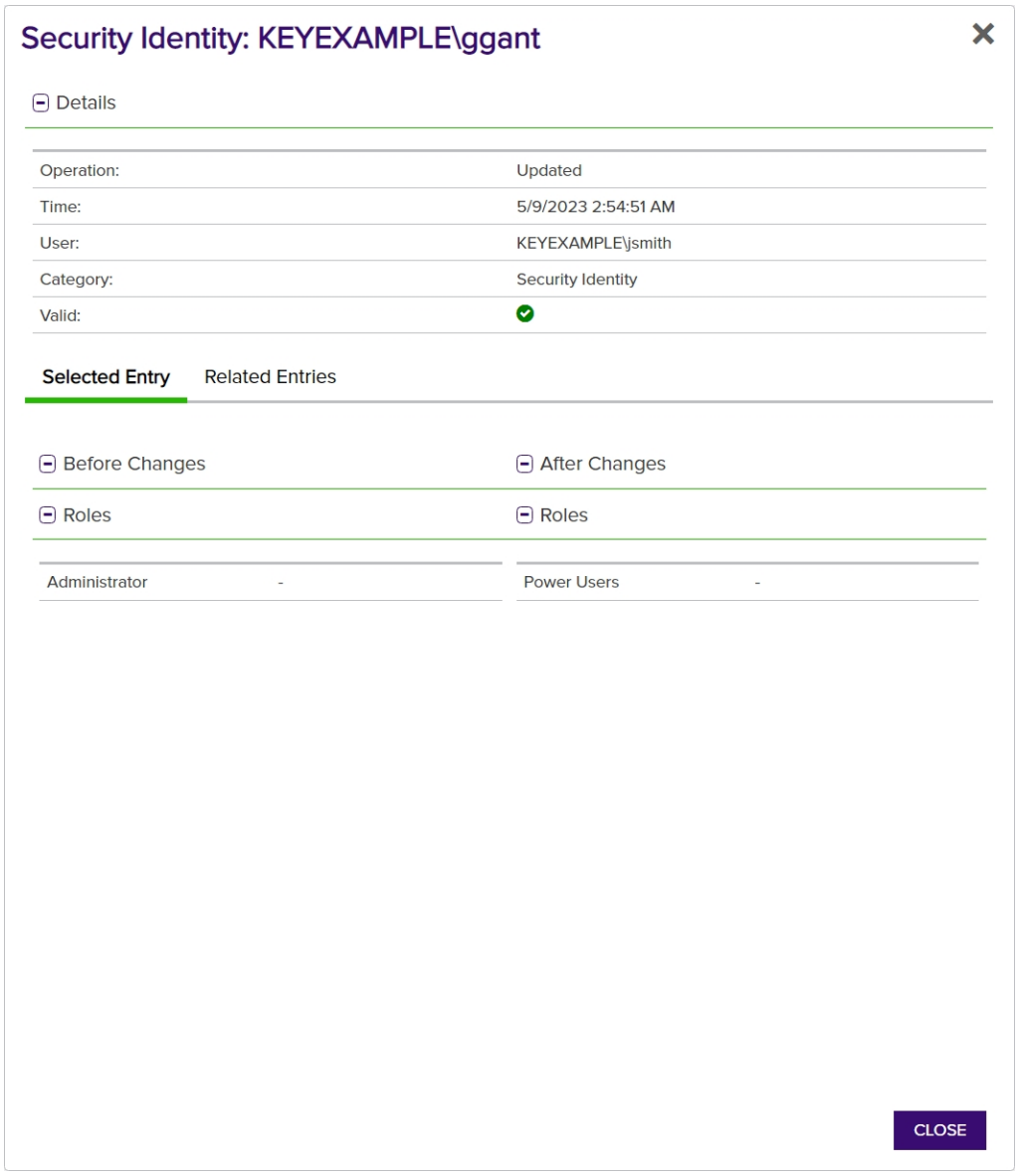


Figure 384: Audit Log Details for Security

SSH

SSH key management with the Keyfactor Bash Orchestrator generates a wide variety of audit log entries, including:

- An SSH user key is created or updated (with an object name of the user)
- An SSH service account key is created, updated, or deleted (with an object name of the service account)
- An SSH key is updated or deleted (with an object name of the public key fingerprint rather than user or service account name—this references the same key as issued for the user or service

account)

- An SSH logon is created, updated, or deleted
- An SSH server is created, updated, or deleted
- An SSH server group is created, updated, or deleted
- A rogue SSH key is identified associated with a logon while scanning a server configured for SSH key management
- An SSH key rotation alert is created, updated or deleted

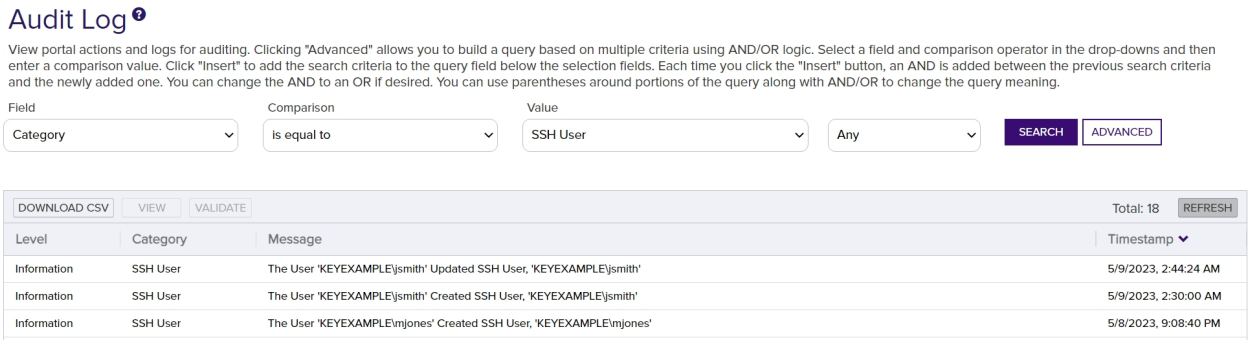


Figure 385: Audit Logs for SSH Management

System Audit Logs

Audit log entries are created during the initial Keyfactor Command installation process when the initial templates and API applications are configured and application settings established. Audit log entries may also be created when you re-run the Keyfactor Command configuration wizard if you make an auditable change in the wizard. When you upgrade from a previous version of Keyfactor Command or make a change in the configuration wizard to an existing Keyfactor Command installation, the audit log entries will show as *Updated*. The exact number of entries created depends on the configuration options selected, number of templates, and the templates configured for enrollment in Keyfactor Command.

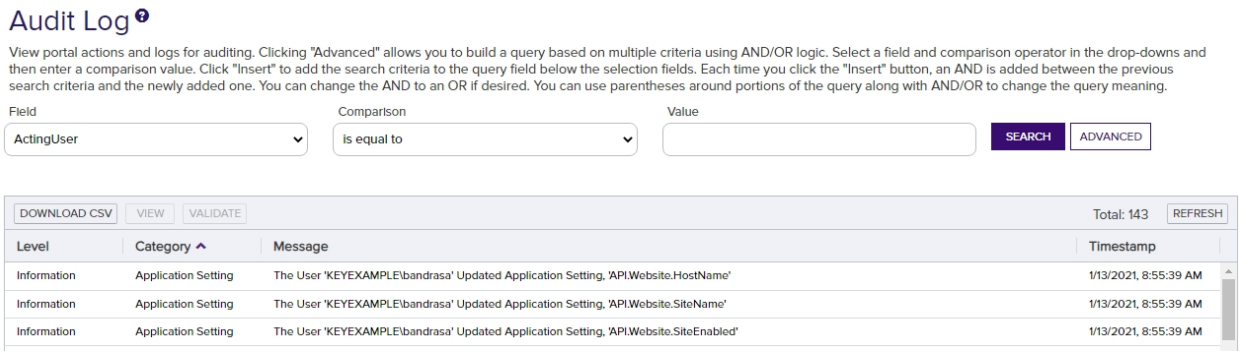


Figure 386: Automated Entries Created by the System in the Audit Log

Workflow

Audit log entries that are generated for workflow include:

- Workflow definition is created
- Workflow definition is imported
- Workflow definition is edited and saved
- Workflow definition is published
- Workflow definition is deleted
- Workflow instance is initiated (created)
- Workflow instance is suspended due to workflow configuration (e.g. the workflow requires approval)
- Workflow instance is stopped manually (this appears as an update to the status of the workflow from Can Receive Signals = True to Can Receive Signals = False)
- Workflow instance is restarted
- Workflow instance failed
- Workflow instance completed

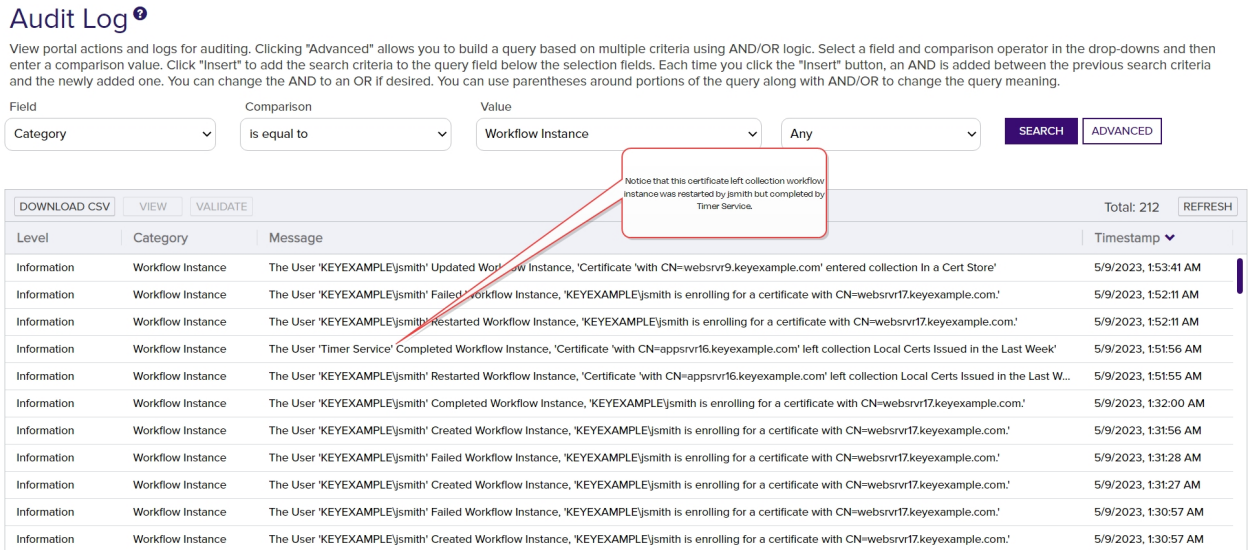


Figure 387: Audit Log Entries for Workflow

For more information about the audit log and using the audit log search feature, see Audit Log on page 716.

Audit Log Security

Keyfactor considers the security and integrity of the audit log to be of the utmost importance and takes steps to ensure that transactions are recorded to the audit log accurately and retained

without tampering until they are purged (by default, after 7 years—see Application Settings: Auditing Tab on page 608).

When Keyfactor Command is installed, a 64-byte key is generated for use in securing audit logs. This key is unique for the implementation. The key is encrypted and stored in the secrets table in SQL using either SQL-level encryption or application-level encryption, depending on the level of encryption selected during installation (see Database Tab on page 2799). If application-level encryption is selected, use of a hardware security module (HSM) is supported. For more information, see Acquire a Public Key Certificate for the Keyfactor Command Server on page 2767.

When an audit log record is created, the key components of it are signed using the unique 64-byte key and stored in the SQL database. The signature is retained and tracked. When the audit log is read, it is validated using the signature. If the signature does not match, the audit log is flagged as invalid (see Validate on page 726), as this could indicate that the record has been tampered with. The following data is included in the key components:

- The date and time at which the action took place.
- The audit message content, which will vary depending on the type of action that was audited. For example, for a modification to a template, this would include:
  - Template common name (short name)
  - Template name
  - Template OID
  - Key size
  - Key type
  - Configuration tenant (forest)
  - Private key retention setting
  - Key archival setting
  - Allowed requesters setting

  See also Download CSV on page 721.
- The operation type (see Audit Log Operations on page 727).
- The user who performed the auditable action.

In order to access the audit logs, users must be granted the **Read** role permission for the **Auditing** role (see Security Roles and Claims on page 622). Users with auditing Read permissions are allowed to access the audit log page and make API requests to obtain data from the audit log.

Important: Be aware that this permission essentially grants a user global read access to the product since the user will be able to view, from the audit log, many of the actions being taken in Keyfactor Command.

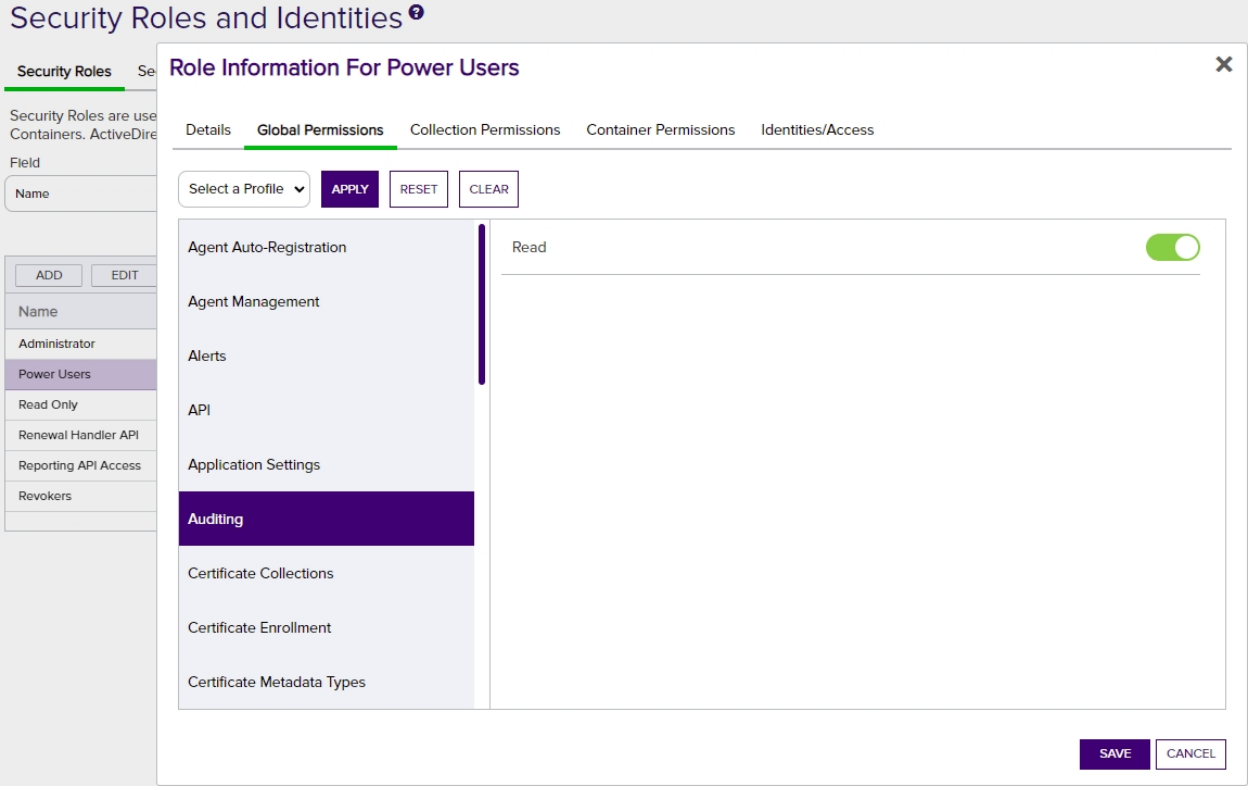


Figure 388: Security Role Showing Auditing Permissions Setting

2.1.11.6 Event Handler Registration

Event handlers are used with expiration and enrollment (pending, issued and denied certificate requests) alerts to trigger additional automated tasks at the time the alerts are run. Keyfactor Command workflows (see Workflow Definitions on page 230) **do not** use event handlers.

Keyfactor provides several event handlers out of the box:

Expiration Logger

Log events to the Keyfactor Command server Windows event log (or other machine event log) for each expiration alert when the alert task is triggered.

Expiration PowerShell

Run a PowerShell script on the Keyfactor Command server for each expiration alert when the alert task is triggered.

Expiration Renewal

Issued PowerShell

Run a PowerShell script on the Keyfactor Command server for each issued certificate alert when the alert task is triggered.

Denied Logger

Log events to the Keyfactor Command server Windows event log (or other machine event log) for each denied request alert when the alert task is triggered.

Denied PowerShell

Execute a certificate renewal for each expiring certificate that is found in a supported certificate store for each expiration alert when the alert task is triggered.

Pending Logger

Log events to the Keyfactor Command server Windows event log (or other machine event log) for each pending request alert when the alert task is triggered.

Pending PowerShell

Run a PowerShell script on the Keyfactor Command server for each pending request alert when the alert task is triggered.

Issued Logger

Log events to the Keyfactor Command server Windows event log (or other machine event log) for each issued certificate alert when the alert task is triggered.

Run a PowerShell script on the Keyfactor Command server for each denied request alert when the alert task is triggered.

SSH Key Rotation Logger

Log events to the Keyfactor Command server Windows event log (or other machine event log) for each SSH key rotation alert when the alert task is triggered.

SSH Key Rotation PowerShell

Run a PowerShell script on the Keyfactor Command server for each SSH key rotation alert when the alert task is triggered.

For information on using built-in event handlers, see Using Event Handlers on page 218.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Custom Event Handler Operations

Custom event handlers are used by expiration and enrollment alerts (see Alerts on page 166) but **not** by Keyfactor Command workflows (see Workflow on page 229).

Registering a Custom Event Handler

The built-in event handlers are registered as part of the Keyfactor Command installation. You should only need to use this option if you have a custom event handler.

To register custom event handlers:

1. In the Management Portal, browse to *System Settings Icon* > *Event Handler Registration.*
2. On the Event Handler Registration page, click **Analyze Handler File**.

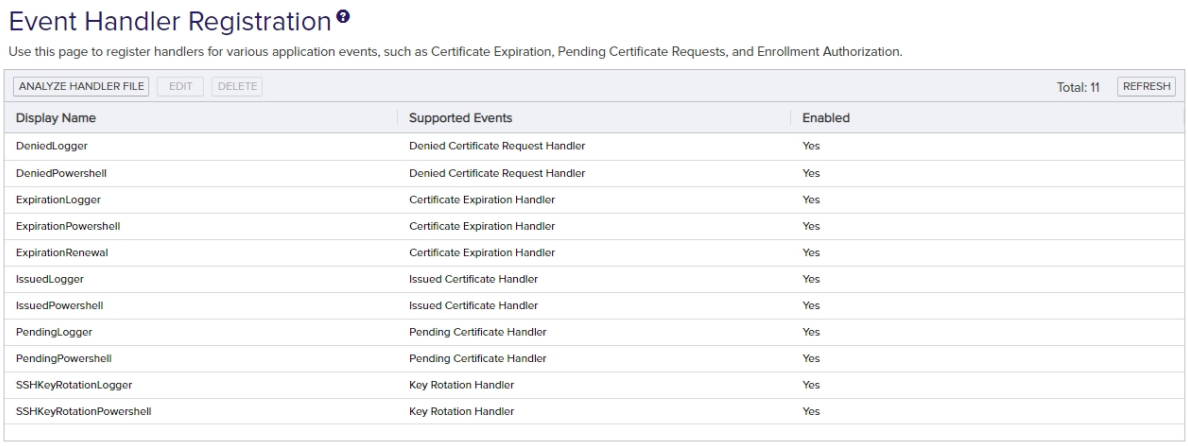


Figure 389: Event Handler Registration Grid

3. In the Analyze Event Handler Assembly File dialog, enter the file name for the event handler file (provided by Keyfactor if the file has been created by Keyfactor) for analysis and click **Save**.

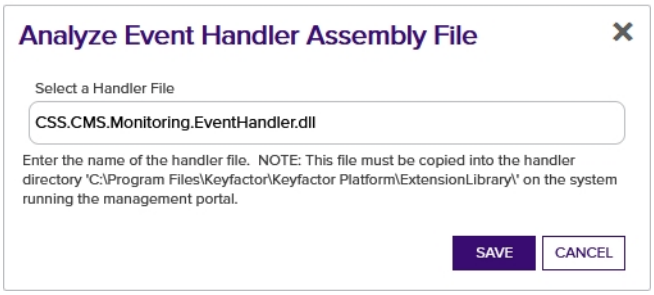


Figure 390: Event Handler Registration

Deleting an Event Handler

To delete an event handler:

1. Browse to *System Settings Icon* > *Event Handler Registration.*
2. Highlight the row in the grid and click **Delete** at the top of the grid.

Editing an Event Handler

To edit an event handler:

1. Browse to *System Settings Icon* > *Event Handler Registration.*
2. Double-click the event handler or highlight the row in the grid and click **Edit** at the top of the grid.
3. In the Event Handler Registration dialog, you can change the **Display Name** for the event handler, if desired. This name appears in the dropdowns in the expiration, pending request, issued certificate, and denied request alert configuration dialogs. You can also disable the event handler by unchecking the **Enabled** box. If you disable an event handler, it will not appear in the dropdowns in the alert configuration dialogs.
4. Click **Save**.

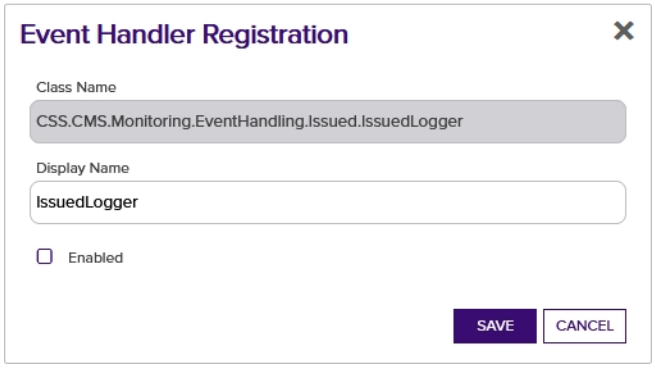


Figure 391: Event Handler Registration Editor

2.1.11.7 Privileged Access Management (PAM)

Privileged access management (PAM) functionality in Keyfactor Command allows for configuration of third party PAM providers to secure certificate stores, credentials for accessing certificate authorities, and similar. PAM functionality is provided using custom PAM extensions. Keyfactor provides several PAM extensions on the publicly-facing Keyfactor GitHub:

https://keyfactor.github.io/integrations-catalog/content/pam

The Keyfactor Command PAM solution is made up of three elements:

- Install an appropriate custom PAM provider extension (see Installing Custom PAM Provider Extensions on the next page).
- Create a PAM provider record in Keyfactor Command (see PAM Provider Configuration in Keyfactor Command on page 749).
- Apply PAM provider security to individual certificate stores (see Adding or Modifying a Certificate Store on page 413), certificate authority records and other locations as needed in Keyfactor Command.

PAM Extensions support installation either locally (on the Keyfactor Command server) or remotely (on each instance of the Keyfactor Universal Orchestrator that will be accessing PAM secrets). You will need to make a determination as to which installation type best meets your needs:

- Local (on the Keyfactor Command server) installations support any type of PAM secret storage supported by Keyfactor Command, including certificate stores and certificate authority secrets, but may require greater accessibility between the Keyfactor Command server and the PAM provider than is desired for your environment.
- Remote (on the orchestrator) installations support PAM secret storage only for the certificate stores managed by the Universal Orchestrator where the PAM extension is installed, but may be a better choice in terms of network accessibility for your environment.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Installing Custom PAM Provider Extensions

Before you can begin to use a third party PAM provider with Keyfactor Command, you need to acquire and install the appropriate custom-built PAM extension for your desired PAM provider from the Keyfactor GitHub:

https://keyfactor.github.io/integrations-catalog/content/pam

To find a package on GitHub:

1. Visit the link above to find your desired package, and click either **Github Repository** or **View source on GitHub** to go to the package page on GitHub.

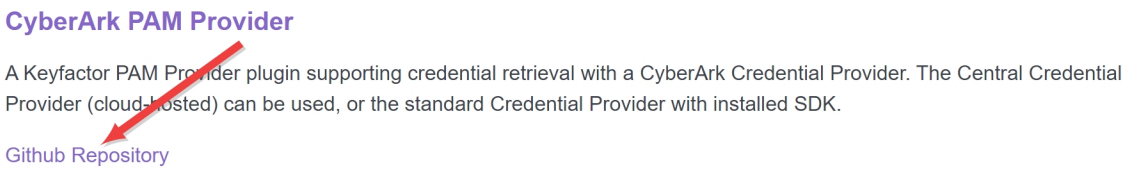


Figure 392: View Packages as Part of a List

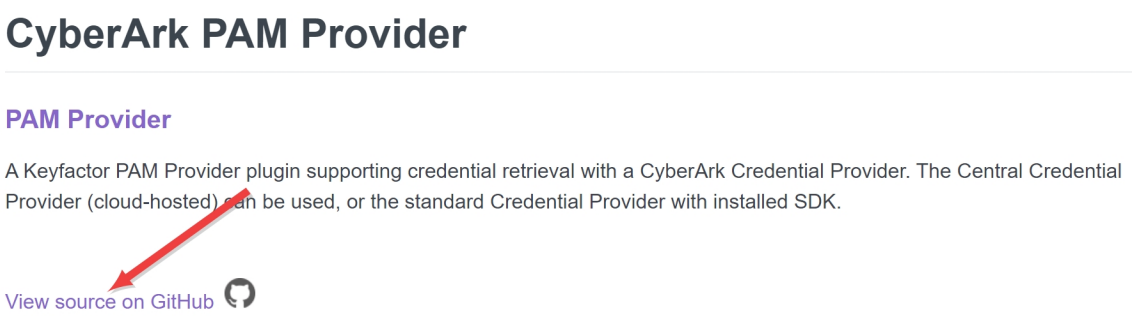


Figure 393: View Packages on Individual Pages

2. On the GitHub page, on the right-hand side, click the link for the **Latest** version.

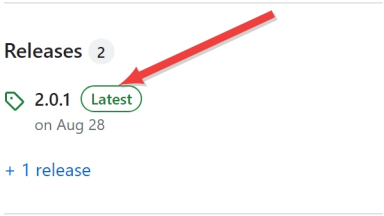


Figure 394: Find the Latest Version of the Package

3. On the GitHub version page in the Assets section, click the package name to download the zip file.

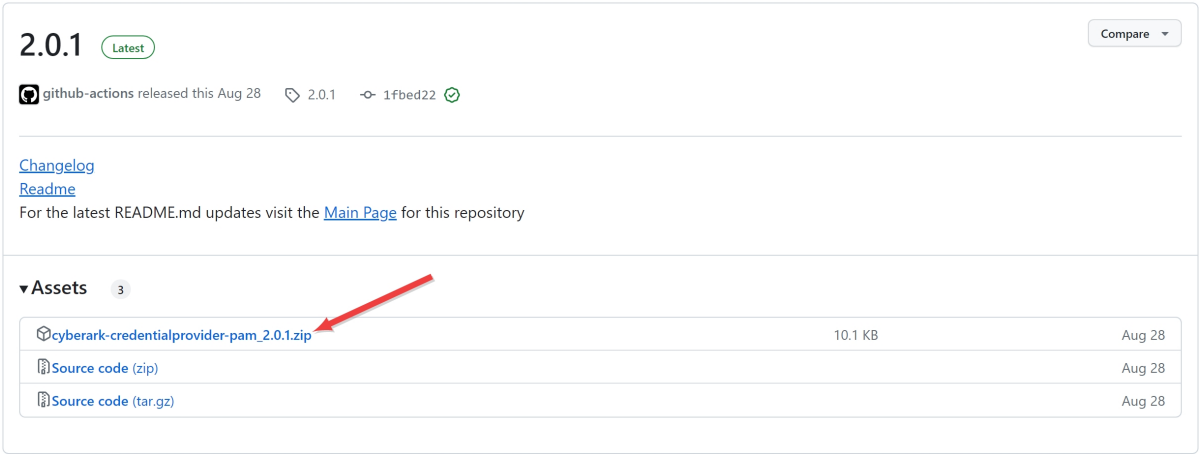


Figure 395: Download the Package Zip File

4. On the main extension GitHub page, review the documentation for the specific extension. Here you will find supported platforms, prerequisites, and extension-specific installation and configuration instructions. The below instructions only cover where to place the extension files on either the Keyfactor Command or the orchestrator and building custom manifest.json files (changes to which aren't needed for extensions from GitHub unless you are customizing something), but not the details for creation of a PAM provider type in Keyfactor Command for the extension or any other customization specific to a given extension.

Extensions support installation either locally (on the Keyfactor Command server) or remotely (on each instance of the Keyfactor Universal Orchestrator that will be accessing PAM secrets). Be sure to follow the installation instructions for the type of installation you wish to do:

- Local (on the Keyfactor Command server) installations support any type of PAM secret storage supported by Keyfactor Command, including certificate stores and certificate authority secrets, but may require greater accessibility between the Keyfactor Command server and the PAM provider than is desired for your environment.
- Remote (on the orchestrator) installations support PAM secret storage only for the certificate stores managed by the Universal Orchestrator where the PAM extension is installed, but may be a better choice in terms of network accessibility for your environment.

Installation on the Keyfactor Command Server

To install a PAM extension on the Keyfactor Command server:

1. Using the Keyfactor API, add a PAM provider type in Keyfactor Command for the custom PAM extension. See the *Adding a PAM Provider Type* instructions for your selected PAM extension. The following is an example request body for a POST /PamProviders/Types request to create a PAM provider type for the Delinea extension:

```
{
   "Name":"Delinea-SecretServer",
   "Parameters":[
      {
         "Name": "Host",
         "DisplayName":"Secret Server URL",
         "InstanceLevel":false,
         "DataType": "string"
      },
      {
         "Name":"Username",
         "DisplayName":"Secret Server Username",
         "InstanceLevel":false,
         "DataType": "secret"
      },
      {
         "Name":"Password",
         "DisplayName":"Secret Server Password",
         "InstanceLevel":false,
         "DataType": "secret"
      },
      {
         "Name":"SecretId",
         "DisplayName":"Secret Server Secret ID",
         "InstanceLevel":true,
         "DataType": "string"
      },
      {
         "Name":"SecretFieldName",
         "DisplayName":"Secret Field Name",
         "InstanceLevel":true,
         "DataType": "string"
      }
   ]
}
```

2. On the Keyfactor Command server, locate the \WebAgentServices\Extensions\PamProviders directory within the install directory. By default, this is:

   C:\Program Files\Keyfactor\Keyfactor Platform\WebAgentServices\Extensions\PamProviders

3. Under the Pam Providers directory, create a new directory with an appropriate name for the PAM extension (e.g. CyberArk). This name is for reference only and does not need to match any names used elsewhere.

4. Place the files you downloaded for the PAM extension in the new directory.

5. In the directory for the PAM extension, locate the file called manifest.json. The manifest.json file should be placed in the same directory as the DLL(s) for the extension.

6. Using a text editor, open the manifest.json file for editing and configure it appropriately for the extension, if needed. See the *Configuring the Server Side manifest.json File* instructions for your selected PAM extension for exact details. The following Delinea manifest.json file is provided as an example. Things to note:

    - Areas highlighted in red and yellow, below, will vary between manifest.json files.
    - When you add your PAM provider into Keyfactor Command, the name you give it must match the name highlighted in red, below (for your manifest.json file).
    - In most cases, server-side files should not need customization.

    ```
    {
       "extensions": {
          "Keyfactor.Platform.Extensions.IPAMProvider": {
             "PAMProviders.Delinea.PAMProvider": {
                "assemblyPath": "delinea-secretserver-pam.dll",
                "TypeFullName": "Keyfactor.Extensions.Pam.Delinea.SecretServerPam"
             }
          }
       }
    }
    ```

7. If you'll be using PAM to store secrets for uses other than certificate stores, repeat the above steps for the PamProviders directories found here, by default:

    C:\Program Files\Keyfactor\Keyfactor Platform\WebConsole\Extensions\PamProviders

    C:\Program Files\Keyfactor\Keyfactor Platform\KeyfactorAPI\Extensions\PamProviders

    Note: Step 1 (add a PAM provider type into Keyfactor Command) does not need to be repeated. The same PAM provider type and PAM provider may be used from multiple areas of the product.

8. Restart the web server services (iisreset) on the Keyfactor Command to complete the implementation.

Installation on the Keyfactor Universal Orchestrator Server

To install a PAM extension on a Universal Orchestrator for use by that orchestrator only:

1. Using the Keyfactor API, add a PAM provider type in Keyfactor Command for the custom PAM extension. See the *Adding a PAM Provider Type* instructions for your selected PAM extension. The following is an example request body for a POST /PamProviders/Types request to create a PAM provider type for the Delinea extension:

```
{
   "Name":"Delinea-SecretServer",
   "Parameters":[
      {
         "Name": "Host",
         "DisplayName":"Secret Server URL",
         "InstanceLevel":false,
         "DataType": "string"
      },
      {
         "Name":"Username",
         "DisplayName":"Secret Server Username",
         "InstanceLevel":false,
         "DataType": "secret"
      },
      {
         "Name":"Password",
         "DisplayName":"Secret Server Password",
         "InstanceLevel":false,
         "DataType": "secret"
      },
      {
         "Name":"SecretId",
         "DisplayName":"Secret Server Secret ID",
         "InstanceLevel":true,
         "DataType": "string"
      },
      {
         "Name":"SecretFieldName",
         "DisplayName":"Secret Field Name",
         "InstanceLevel":true,
         "DataType": "string"
      }
   ]
}
```

2. On the Universal Orchestrator server, locate the extensions directory within the install directory. By default, this is:

   C:\Program Files\Keyfactor\Keyfactor Orchestrator\extensions

3. Under the extensions directory, create a new directory with an appropriate name for the PAM extension (e.g. CyberArk). This name is for reference only and does not need to match any names used elsewhere.

4. Place the files you downloaded for the PAM extension in the new directory.

5. In the directory for the PAM extension, locate the file called manifest.json. The manifest.json file should be placed in the same directory as the DLL(s) for the extension.

6. Using a text editor, edit the manifest.json file and configure it appropriately for the extension. See the *Configuring the Client Side manifest.json File* instructions for your selected PAM extension for exact details. The following Delinea manifest.json file is provided as an example. Things to note:

    - Areas highlighted in red and yellow, below, will vary between manifest.json files.
    - Areas shown in red text are examples of items that need to be customized for your environment.
    - When you add your PAM provider into Keyfactor Command, the name you give it must match the name highlighted in red, below (for your manifest.json file).

```
{
   "extensions": {
      "Keyfactor.Platform.Extensions.IPAMProvider": {
         "PAMProviders.Delinea-SecretServer.PAMProvider": {
            "assemblyPath": "delinea-secretserver-pam.dll",
            "TypeFullName": "Keyfactor.Extensions.Pam.Delinea.SecretServerPam"
         }
      }
   },
   "Keyfactor:PAMProviders:Delinea-SecretServer:InitializationInfo": {
      "Host": "http://127.0.0.1:8200",
      "Path": "v1/secret/data",
      "Token": "xxxxxx"
   }
}
```

7. Restart the Universal Orchestrator service (see Start the Universal Orchestrator Service on page 2953).

PAM Provider Configuration in Keyfactor Command

Any third-party privilege access management (PAM) providers you wish to configure for use with Keyfactor Command must be defined first on the PAM Providers page before they can be assigned to certificate stores (see Certificate Stores on page 408), used for explicit credentials on a CA (see Adding or Modifying a CA Record on page 354), or used to provide authentication in workflow steps

(see). Keyfactor Command supports multiple custom-built PAM providers are available on the Keyfactor GitHub:

https://keyfactor.github.io/integrations-catalog/content/pam

PAM providers can either be local (server side) or remote (client side). When configured locally, the configuration information to connect to the PAM provider exists on the Keyfactor Command server and the PAM provider must be routable from the Keyfactor Command server (for example, on the same network) to retrieve secret information. When configured remotely, the configuration information to connect to the PAM provider exists on the Keyfactor Universal Orchestrator managing the certificate stores using the PAM provider and the PAM provider must be routable from the Universal Orchestrator.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
PAM > Modify
PAM > Read
Certificate Stores > Modify

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Adding or Modifying a PAM Provider

The PAM provider configuration can be edited at any time, even if it is used on existing records.

To define a new PAM provider or modify an existing one:

1. In the Management Portal, browse to *System Settings Icon* > *Privileged Access Management.*
2. On the PAM Providers page, click **Add** to create a new provider, or, to modify an existing provider, double-click the provider, right-click the provider and choose **Edit** from the right-click menu, or highlight the row in the providers grid and click **Edit** at the top of the grid.
3. In the PAM Providers dialog, check the **Remote Provider** box if you are adding a PAM provider for a PAM extension installed on a Universal Orchestrator.

   A remote PAM provider generally exists outside the local network of the Keyfactor Command server. This option allows you to specify the secret information in Keyfactor Command in the same way as you would with a local PAM provider without needing to enter PAM provider configurations in Keyfactor Command (other than a base remote provider link). The PAM provider configuration information is, instead, supplied in the orchestrator's PAM manifest (see Installing Custom PAM Provider Extensions on page 743). Remote PAM providers are only supported for use with certificate stores and the Keyfactor Universal Orchestrator version 10.0 or greater.

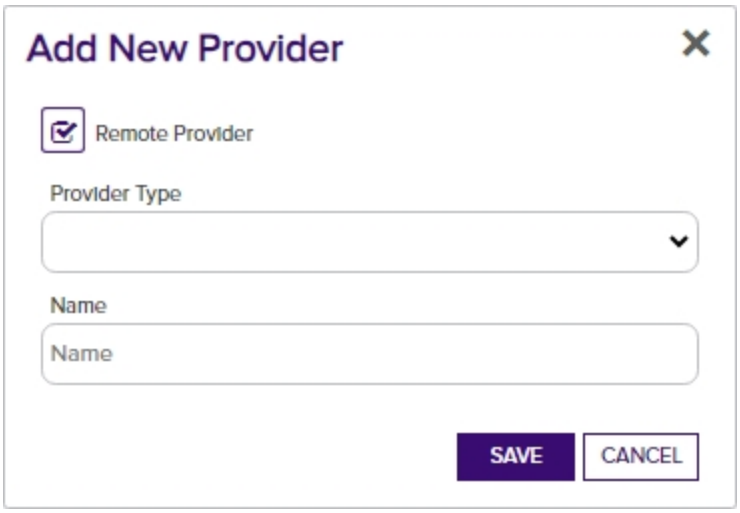


Figure 396: Remote PAM Provider

4. Select a **Provider Type** in the dropdown. This is the name of the software vendor that provides your Privilege Access Management solution. This field cannot be modified on an edit.

   Note: If a provider type does not already exist for the PAM provider you are adding, you will need to create a new supported type before completing this step (see Installing Custom PAM Provider Extensions on page 743).

5. In the **Name** field, enter the name for the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.

   Important: The name you give to your PAM provider in Keyfactor Command must match the name of the PAM provider as referenced in the manifest.json file (see Installing Custom PAM Provider Extensions on page 743).

6. The remainder of the fields in the dialog will vary depending on the provider type selected. If you checked Remote Provider, no further configuration is needed in this dialog. For example:

   CyberArk Extension (Central Credential Provider)

   - **Application ID**: The name/ID of the application created for Keyfactor Command.
   - **CyberArk Host and Port**: The hostname or IP address where CyberArk is hosted, including port. Do not include http/https.
   - **CyberArk API Site**: The web server site name to which CyberArk has been deployed. By default, this is AIMWebService.

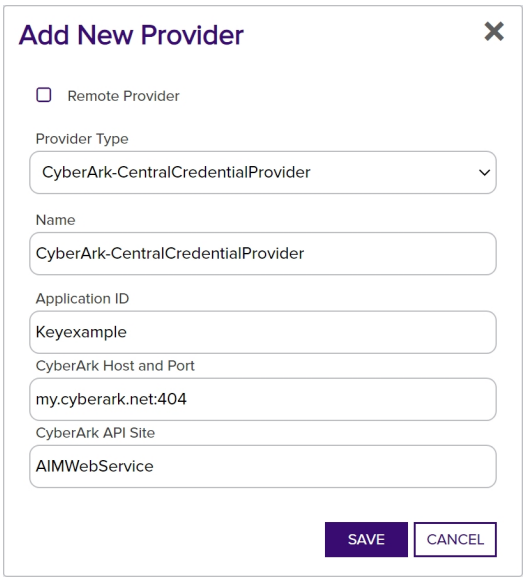


Figure 397: Create CyberArk Provider

Delinea Extension

- **Secret Server URL**: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://websrvr38.keyexample.com/SecretServer).
- **Secret Server Username**: The username of the user that will be used to connect to SecretServer.
- **Secret Server Password**: The password of the user that will be used to connect to SecretServer.

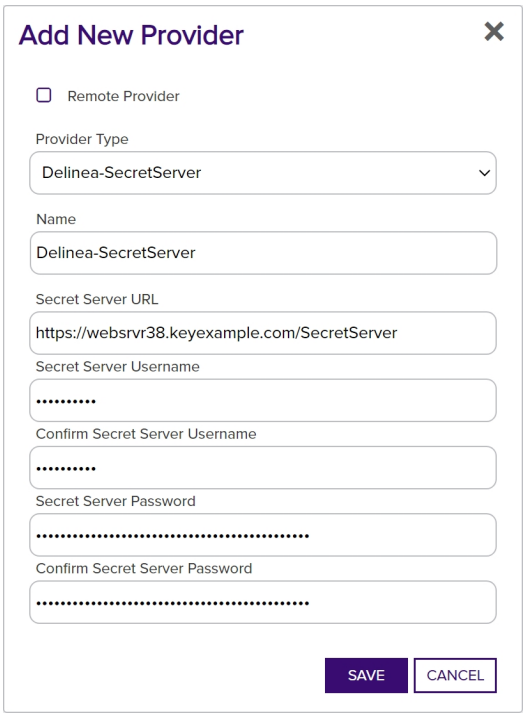


Figure 398: Create Delinea PAM Provider

HashiCorp Extension

- **Vault Host**: The URL to the vault instance, including port number if applicable (e.g. https://websrvr35.keyexample.com:8200).
- **Vault Token**: The access token for the vault (assuming Kerberos authentication is not used).
- **KV Engine Path**: The path to the vault secrets. The default is *v1/secret/data*.

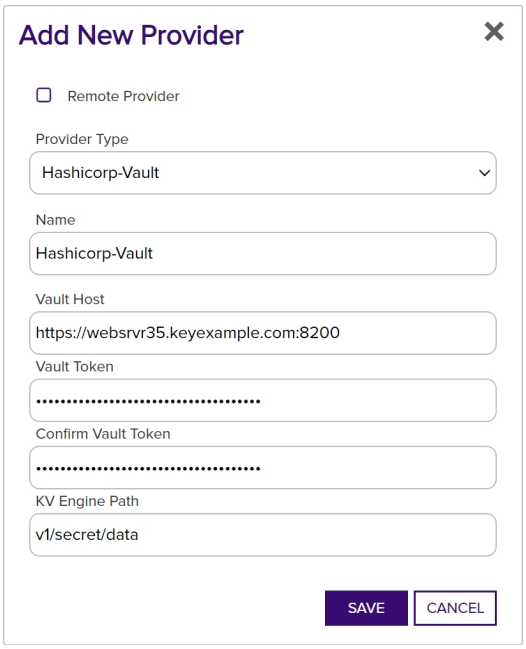


Figure 399: Create HashiCorp PAM Provider

7. Click **Save** to save the provider.

Deleting a PAM Provider

To delete a provider, highlight the row in the providers grid and click **Delete** at the top of the grid or right-click the provider in the grid and choose **Delete** from the right-click menu.

Tip: If a PAM provider has been associated with any certificate stores or CAs, it cannot be deleted.

2.1.11.8 Identity Providers

Identity providers in Keyfactor Command are used to provide a method for authenticating access to the Keyfactor Command Management Portal and Keyfactor API. An identity provider for Keyfactor Command is selected during the installation process and typically would not need to be updated. However, if you decide to migrate your Keyfactor Command implementation from one identity provider to another, you can add the new provider to which you will migrate using the options in this interface. Authentication with only one identity provider at a time for a given Keyfactor Command server is supported.

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.

You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

Identity Provider Operations

On the identity providers page, you can modify existing identity providers, but you cannot add new identity providers. New identity providers should be added by re-running the Keyfactor Command Configuration Wizard and adding a new identity provider on the Authentication tab (see Authentication Tab on page 2787). Identity providers cannot be deleted.

To modify an identity provider:

1. In the Management Portal, browse to *System Settings Icon* > *Identity Providers*.
2. On the Identity Providers page, highlight a row and click **Edit** from the top of the grid or from the right click menu to modify an existing provider.
3. On the Editing Identity Provider page, fill in each tab of the dialog with the information desired for the selected identity provider.
   a. On the Details tab, enter a short *Name* and *Display Name* for the provider. The *TypeId* cannot be edited.

      Important: The value in the Name field must match the provider name referenced in the redirect URLs (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

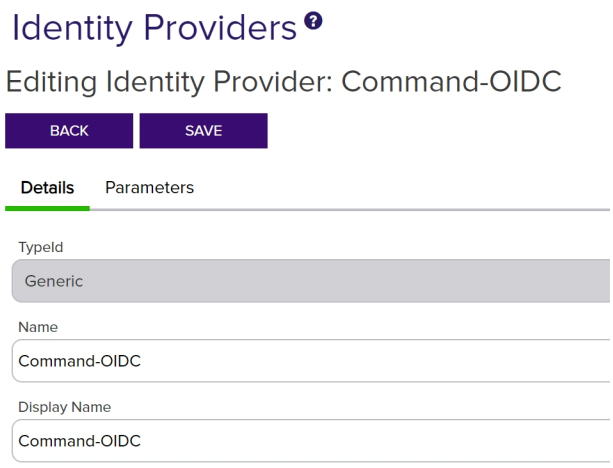
      

      Figure 400: Details for an Identity Provider

   b. On the Parameters tab, select each parameter to configure and click **Edit** to open the Edit <Parameter Name> Parameter dialog, the contents of which will vary depending on the

parameter selected. For information about the specific parameters, see Table 76: Identity Provider Parameters.

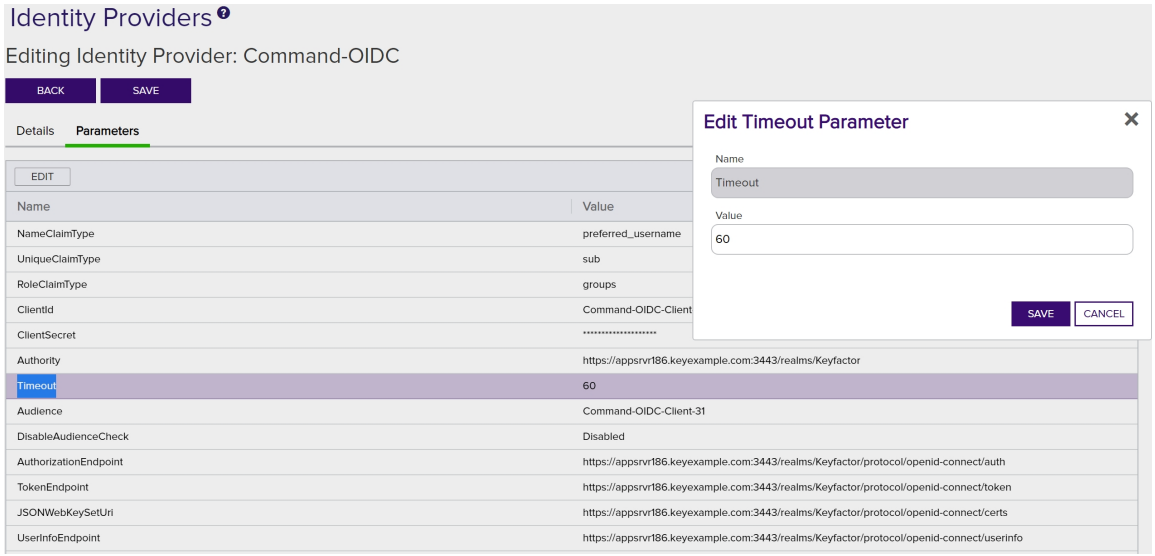


Figure 401: Edit Parameters for an Identity Provider

4. Click **Save** to save the role.

Table 76: Identity Provider Parameters

Name	Type	Example	Description
Admin Querying Client Id	1 - String	Command-API-Query	The client ID of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). Keyfactor recommends that you use a different client for this purpose than the client used for the main connection from Keyfactor Command to the identity provider (see *Client Id*). This parameter is required.
Admin Querying Client Secret	1 - String		The client secret of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). This parameter is required.

Name	Type	Example	Description
Audience	1 - String	Command-OIDC-Client	The audience value for the identity provider. For Keyfactor Identity Provider, this should be set to the same value as the *Client Id*. For example: `Command-OIDC-Client` This parameter is required.
Auth0 API URL	1 - String		The unique identifier defined in Auth0 or a similar identity provider for the API. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Authority	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor	The issuer/authority endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required. **Tip:** When you add or update an identity provider, the provider's discovery document is validated based on this authority URL. The discovery document is also validated periodically in the background. The following are validated: • That the discovery document is reachable using the Authority value provided and can be parsed into a valid discovery

Name	Type	Example	Description
			document. • That the Authority URL matches the Issuer returned in the discovery document. • That all the URLs on the discovery document are using HTTPS. • That the JSONWebKeySetUri value is included on the discovery document. • That any endpoint configuration values (Authorization Endpoint, Token Endpoint, UserInfo Endpoint, JSONWebKeySetUri) that have been saved or are being saved match—including case—the values returned in the discovery document. The UserInfo Endpoint is not a required configuration field, but if a value is provided, it must match what's in the discovery document. If any of these validation tests fail, any identity provider changes in process will not be saved and an error will be displayed or logged.
Authorization Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/auth	The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery*

Name	Type	Example	Description
			Document Endpoint **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Client Id	1 - String	Command-OIDC-Client	The ID of the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, this should be: `Command-OIDC-Client` For more information, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). This parameter is required.
Client Secret	2 - Secret		The secret for the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716 for help locating this. It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information.

Name	Type	Example	Description
			This parameter is required.
Discovery Document Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/.well-known/openid-configuration	The discovery URL for the identity provider. For Keyfactor Identity Provider, this is the link to the OpenID Endpoint Configuration page, which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). Populate this value and click **Fetch** to populate the remainder of the fields in this section, if desired. If you opt not to populate this field or if the discovery document does not return a valid response, the remainder of the fields in this section of the configuration will need to be configured manually. This value is not stored in the database.
Fallback Unique Claim Type	1 - String	cid	A backup value used to reference the type of claim used for users in the identity provider in case the primary referenced name does not contain a value. This parameter is required.
JSON Web Key Set Uri	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/certs	The JWKS (JSON Web Key Set) URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Name Claim Type	1 - String	preferred_username	The name used to reference the type of user claim for the identity provider.

Name	Type	Example	Description
			For Keyfactor Identity Provider, this should be: `preferred_username` This parameter is required.
Role Claim Type	1 - String	groups	The value used to reference the type of group claim for the identity provider. For Keyfactor Identity Provider, this should be: `groups` This parameter is required.
Scope	1 - String		One or more scopes that are requested during the OIDC protocol when Keyfactor Command is the relying party. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Sign Out URL	1 - String	https://my-auth0-instance.us.auth0.com/oidc/logout	The signout URL for the identity provider. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Timeout	1 - String	60	The number of seconds a request to the identity provider is allowed to process before timing out with an error.
Token Audience	1 - String		An audience value to be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. This value is not used for Keyfactor Identity Provider.
Token Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token	The token endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can

Name	Type	Example	Description
			be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Token Scope	1 - String		One or more scopes that should be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Unique Claim Type	1 - String	sub	The value used to reference the type of claim used for users in the identity provider. For Keyfactor Identity Provider, this should be (for subject): `sub` This parameter is required.
User Info Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/certs	The user info endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct.
User Query	1 -	https://my-keyidp-serv-	The user query endpoint URL for the iden-

Name	Type	Example	Description
Endpoint	String	er.keyexample.com /admin/realms/Keyfactor	tity provider. For Keyfactor Identity Provider, this is the base URL for the Keyfactor Identity Provider REST API and has an expected value of: `https://<host>/admin/realms/<realm_name>` This parameter is required.

Using the Identity Provider Search Feature

The search function allows you to query the database for information. The same query structure is used in multiple locations within the Keyfactor Command Management Portal.

When you first open the page, you will see the simple search option. To execute a search, select the field and comparison operators in the dropdowns and type something on which to search in the value field (if applicable). If you select an *is null* or *is not null* comparison operator, the value field will be grayed out. Click the **Search** button to execute the query.

Each query consists of three parts:

Query Field

The available fields for querying vary depending on the area of the Management Portal in which the search is used. On this page, the queries can be done on the following built-in fields:

DisplayName

Complete or partial matches with the display name of the identity provider.

Name

Complete or partial matches with the name of the identity provider.

Private

Whether the identity provider is marked as private, true/false.

ProviderType

Complete or partial matches with the provider type for the identity provider. Supported provider types are:

- Active Directory
- Azure
- Generic

Comparison Operator

The query comparison operators vary depending on the type of field selected and the specific properties of the field. The list below shows the dropdown list comparison operators, as well as the equivalent query language syntax (in parentheses).

Most string fields (the vast majority of the built-in fields) support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Contains (-contains)
- Does not contain (-notcontains)
- Starts with (-startswith)
- Ends with (-endswith)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most date and integer fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is less than (-lt)
- Is less than or equal to (-le)
- Is greater than (-gt)
- Is greater than or equal to (-ge)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Most Boolean (true/false) fields support:

- Is equal to (-eq)
- Is not equal to (-ne)
- Is null (-eq NULL)
- Is not null (-ne NULL)

Comparison Value

The value you enter for comparison must match the field type. For example, integer fields only support numerical values. String fields support all alphanumeric characters. Boolean fields only support True or False. The value field is not case sensitive. Date fields support only properly formatted dates and will initially display as mm/dd/yyyy. You can choose to populate the date field by:

- Clicking in a date Value field to open a pop-up calendar to select a date that will populate the field.
- Clicking in a segment of the date format (i.e., mm/dd/yyyy) and entering a value. As you continue to type in any one segment, the cursor will keep moving onto the next segment.

The results that match your search criteria will be displayed in the results grid below the search selection options.

Advanced Searches

On any search page you can click **Advanced** to the right of the Search button to display the advanced search options. Click **Simple** to close the advanced search options again.

Multiple Criteria

Using the advanced search options, you can build a query based on multiple criteria using AND/OR logic. As with a simple search, you select a field and comparison operator in the drop-downs and

then enter a comparison value, if applicable. Click **Insert** to add the search criteria to the query field below the selection fields. Use the selection fields to build multiple search criteria. Each time you click the insert button, an AND is added between the previous search criteria and the newly added one. You can change the AND to an OR if desired. You can use parentheses around portions of the query along with AND/OR to change the query meaning.

For example, for certificate searches:

```
(CN -contains "appsrvr" AND IssuedDate -ge "01/01/2022") OR (CN -contains "appsrvr" AND
TemplateShortName -contains "web")
```

This query will return all the certificates issued on or after January 1, 2022 with the string appsrvr in the CN and also all certificates issued at any time with the string appsrvr in the CN using a template referencing web. When you have entered all the desired search criteria, click **Search** to execute the query. If you wish to clear the query field and start over, click the **Clear** button.

2.1.11.9 SMTP Configuration

SMTP settings to enable Keyfactor Command to deliver reports and alerts via email are generally specified during initial Keyfactor Command installation and configuration, but can be modify through the Management Portal if needed.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
Security > Modify
Security > Read/

To make a change to these settings:

1. In the Management Portal, browse to *System Settings Icon* > *SMTP Configuration.*
2. On the SMTP Configuration page, modify the configuration as needed.

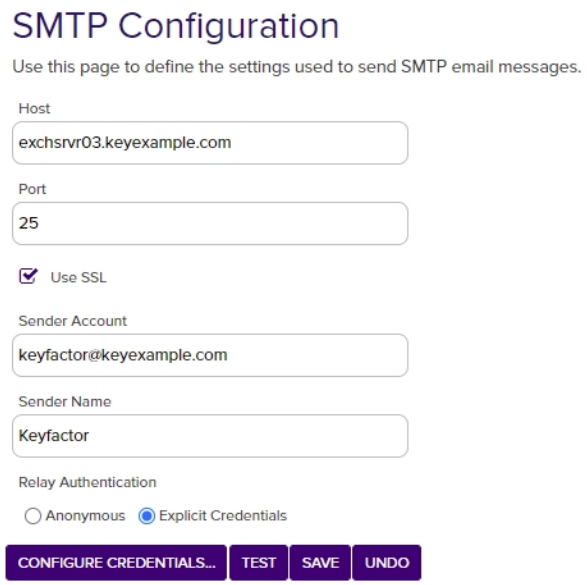


Figure 402: SMTP Configuration

3. Enter the FQDN of your SMTP server in the **Host** field.

4. Enter the SMTP port (default is 25) in the **Port** field.

5. Check the **Use SSL** box if this option is supported by your mail server. Your mail server may not be configured to support TLS/SSL.

6. Set the **Sender Account** name in the form of an email address (e.g. user@keyexample.com). Depending on the email configuration in your environment, the sender account may need to be a valid user on your mail server or you may be able to put anything in this field.

7. Set the **Sender Name** as desired. This is the name that appears as the ”from” in the user’s mail client both with anonymous authentication and explicit credentials.

8. Select the appropriate authentication method for your environment. Some mail servers will accept anonymous. Others may not. If your mail server requires that you provide a username and password for a specific valid user, select the **Explicit Credentials** radio button and click **Configure Credentials**. Enter the valid user’s Active Directory username and password in DOMAIN\username format in the Configure SMTP Relay Authorization Settings dialog. For most mail server configurations, the user you select here must have as a valid email address the email address you set in the *Sender Account* field.

9. You may test the settings prior to saving them. To test the SMTP settings, click the **Test** button, enter a valid email address for a mailbox you can open in the **Send a Test SMTP Message** dialog and click **Send**. Verify that the test email is delivered.

Figure 403: Send an SMTP Test Message

10. Click **Save** to save any changes you have made.

To cancel any changes you've made without saving, click the **Undo** button.

2.1.11.10 Component Installations

On the Component Installations page you can view the components installed on each of your Keyfactor Command servers and, optionally, delete a server if it has been removed from service. The server Hostname and Database name of the Keyfactor Command instance are found on this page.

To delete a server, highlight the row in the component grid and click **Delete** at the top of the grid or right-click the row in the grid and choose **Delete** from the right-click menu. Servers should not be deleted if they are serving any active role in the Keyfactor Command environment, as this operation cannot be undone.

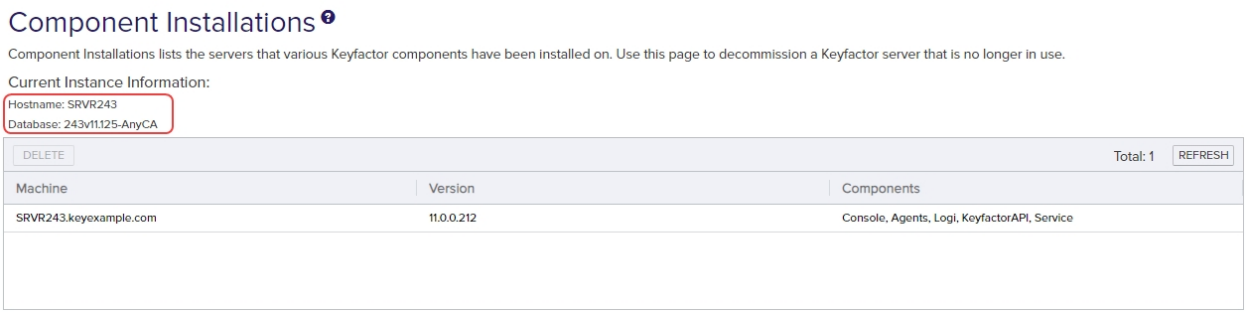


Figure 404: Component Installations

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

2.1.11.11 Licensing

In the Licensing section of the Management Portal you can view the details of your existing license and replace it with a new license, if desired.

To view your existing license, browse to *System Settings Icon* > *Licensing.* The license shows you the features that are enabled for your Keyfactor Command implementation.

For information on monitoring for license expiration, see License Expiration Monitoring and Rotation on page 817.

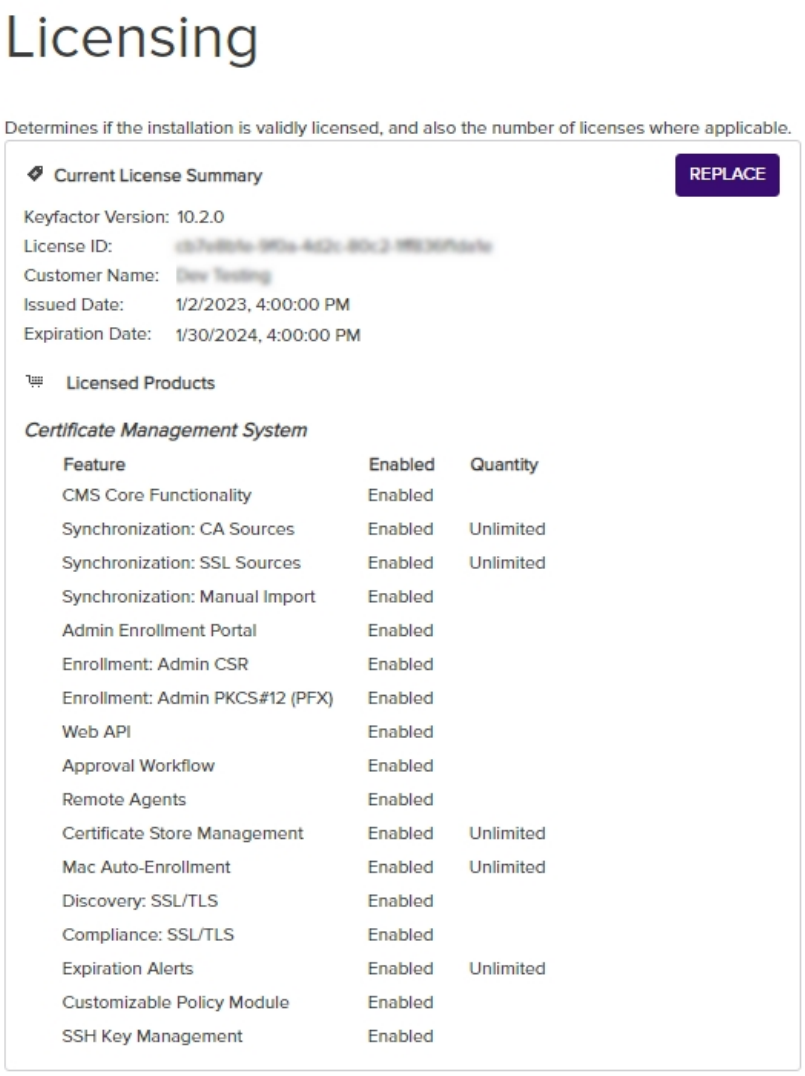


Figure 405: Keyfactor Command License

If you purchase a new license from Keyfactor that enables additional features or extends the expiration date, you can upload it on the Licensing page. To do this:

1. In the Management Portal, browse to *System Settings Icon* > *Licensing*.
2. On the Licensing page, click **Replace**. The Confirm Operation dialog box will open.
3. Click **OK** to open the dialog to upload a new license.

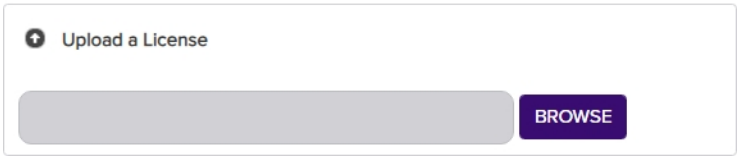


Figure 406: Upload a New Keyfactor Command License

4. Click the **Browse** button and browse to the location on the file system where the new license file provided by Keyfactor is stored.
5. The new license will appear next to the existing license. Compare them to confirm that you wish to install the new license and then click the **Save** to button to complete the license change.

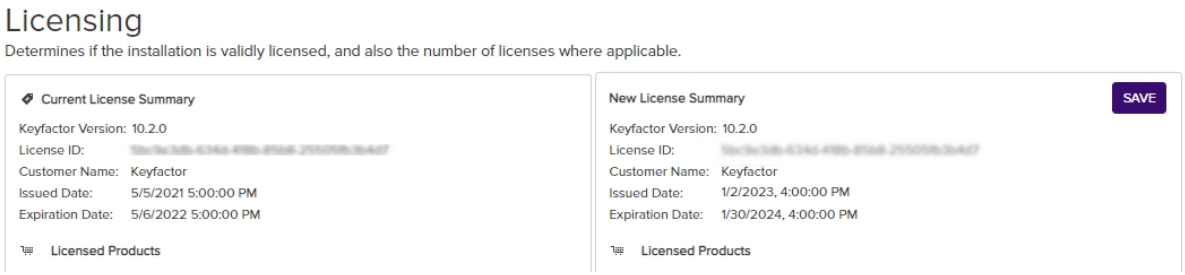


Figure 407: Save a New Keyfactor Command License

6. On the Keyfactor Command server, restart the IIS services (iisreset) and refresh the browser.

Important: If you are installing a new license because your existing license is expiring and you use the Keyfactor CA Policy Module, be aware that the license needs to be installed separately for the policy module (see License Expiration Monitoring and Rotation on page 817).

Tip: Click the help icon (?) next to the page title to open the embedded web copy of the *Keyfactor Command Documentation Suite* to this section.
You can also find the help icon at the top of the page next to the **Log Out** button. From here you can choose to open either the *Keyfactor Command Documentation Suite* at the home page or the *Keyfactor API Endpoint Utility*.

2.2 Operations

Once your Keyfactor Command implementation is up and running, there are a few things you should do regularly to make sure that it stays that way, including backing up to prepare for disaster recovery and monitoring logs and alerts to identify potential issues early and have an overall sense of the health of your Keyfactor Command environment. This section includes information on these things along with some reference information and guidance for optional configuration.

2.2.1 Extending and Customizing Keyfactor Command

There are several options for adding to your Keyfactor Command implementation with custom extensions, handlers, and scripts. There are also a few options for customizations. This section provides an overview of some of the available options.

- PowerShell scripts can be executed from workflows and using event handlers in alerts. For more information, see PowerShell Scripts on page 219.
- The Keyfactor Universal Orchestrator supports the use of custom-built extensions to extend functionality to a variety of certificate store types and devices for management (see Installing Custom-Built Extensions on page 2940).
- The Keyfactor Universal Orchestrator supports the option to implement custom-built certificate store jobs using one or more scripts (PowerShell or Bash) rather than a full extension (see Configuring Script-Based Certificate Store Jobs on page 2946).
- Custom event handlers can be built for use with alerts (see Custom Event Handler Operations on page 740).
- The Keyfactor AnyCA Gateway REST and AnyCA Gateway (previous version) support the use of publicly available extensions to allow for functions such as certificate enrollment and management from Keyfactor Command to a variety of third-party CA vendors (e.g. DigiCert, Entrust, GoDaddy). For more information, see the separate gateway documentation and the Keyfactor GitHub:

  Keyfactor AnyGateway (Older Version)

  Keyfactor AnyCA Gateway REST

  https://keyfactor.github.io/integrations-catalog/content/ca-gateway

- The Keyfactor Command logo on the banner at the top of the Management Portal can be replaced with an alternate image of your choosing (see Customize the Management Portal Banner Logo on page 772).
- Customizations can be done to orchestrator API configuration settings, SQL connection settings, Keyfactor Command Service job settings and more using appsetting.json files (see Keyfactor Command Appsetting.json Files on page 773).

- The executable used to run the Keyfactor Command Service can be changed from an exe to a signed dll for environments where this is an important requirement (see Keyfactor Command Service Executable on page 788).
- Microsoft CA key recovery can be configured on the Keyfactor Command to allow private keys archived in a Microsoft CA to be retrieved in Keyfactor Command (see Configuring Key Recovery for Keyfactor Command on page 789).
- Client certificates used for orchestrator authentication can be renewed using a client certificate renewal extension (see Register a Client Certificate Renewal Extension on page 2961).
- Orchestrators can be auto-registered to Keyfactor Command using a custom auto-registration handler (see Custom Auto-Registration Handlers on page 495).
- At the conclusion of orchestrator jobs a custom handler can be run (see Editing Job Completion Handlers below).
- Privileged Access Management (PAM) providers can be configured either on the Keyfactor Command server or the Keyfactor Universal Orchestrator (see Installing Custom PAM Provider Extensions on page 743).

2.2.1.1 Editing Job Completion Handlers

Job completion handlers are used to run custom handlers at the conclusion of orchestrator jobs. They are configured by placing your custom-built handler in the JobCompleteHandlers directory or a subdirectory of it and editing the manifest.json file in the above directory to appropriately reference your handler.

Keyfactor Command ships with one built-in job completion handler, which is designed to send an email at the conclusion of orchestrator jobs. To configure this handler:

1. First, determine which types of jobs you wish to send notifications about upon job completion and to whom the notifications should be directed. You will need the job type GUIDs (not job GUIDs) for these jobs and the email addresses for the users or groups to whom the notifications will be sent. To determine the job type GUIDs for your jobs, use the GET /CertificateStoreTypes API method (see GET Certificate Store Types on page 1546) and look in the output for the *InventoryJobType*, *ManagementJobType*, and *DiscoveryJobType* GUIDs for each certificate store type you wish to monitor. These are the GUIDs to reference from the job completion handler.

2. On the Keyfactor Command server, locate the \WebAgentServices\Extensions\JobCompleteHandlers directory within the install directory. By default, this is:

   C:\Program Files\Keyfactor\Keyfactor Platform\WebAgentServices\Extensions\JobCompleteHandlers

3. In the JobCompleteHandlers directory, locate the manifest.json file and open it for editing using a text editor.

4. Update the manifest.json file to contain a reference to the SendEmailOrchestratorJobCompleteHandler handler with configuration as shown below.

In the Options section, enter a comma-separated list of the users or groups who should receive the notifications that jobs are complete in the *EmailRecipients*. Enter a comma-separated list of the job type GUIDs (not job GUIDs) that should initiate notifications in the *JobTypes*.

To reduce the number of assemblies listed in the shared assemblies file, add the new extension registration option 'LoadInUpstreamContext' that when set to *True*, causes all of the extension's assemblies to be shared with the calling assembly.

```
{
   "extensions": {
      "Keyfactor.Platform.Extensions.IOrchestratorJobCompleteHandler": {
         "SendEmailOrchestratorJobCompleteHandler": {
            "assemblypath": "../../CSS.CMS.Agents.Server.dll",
            "TypeFullName": "CSS.CMS.A-
gents.Server.Handlers.SendEmailOrchestratorJobCompleteHandler",
            "Options": {
               "EmailRecipients": "john.smith@keyexample.com,martha.jones@keyexample.com",
               "JobTypes": "d60ff0ad-448c-4498-bd73-a8d8194c73c5,b98621f3-d779-40d3-8f09-
eecf32d68183,106478fa-6cbf-4d05-ad31-addaa9675c3c"
            },
            "LoadInUpstreamContext": "True"
         }
      }
   }
}
```

5. Save the manifest.json file.
6. Restart the web server services (iisreset) on the Keyfactor Command server to complete the implementation.

When a job completion email is delivered, it will contain a subject referencing the type of job and that it is complete (e.g. Keyfactor Orchestrator CitrixAdcInventory Job Complete) with a message similar to:

```
CitrixAdcInventory job with id '62c40c40-7904-49be-b3f5-4d49f6856ef2' completed with result: Success.
Job was requested on Wednesday, September 27, 2023 for client machine 'websrvr21.keyexample.com'.
```

2.2.1.2 Customize the Management Portal Banner Logo

You can replace the Keyfactor logo at the top left of the Management Portal with your logo, or any .png image, to customize the appearance for your users. The new image will be displayed across the product for every user accessing the Management Portal. This cannot be selectively applied.

To replace the Keyfactor logo:

1. In Windows Explorer, navigate to the \WebConsole\wwwroot\images directory under the directory in which Keyfactor Command is installed. By default, this is:

   ```
   C:\Program Files\Keyfactor\Keyfactor Platform\WebConsole\wwwroot\images
   ```

2. Rename the Keyfactor *Banner.png* file to *Banner-original.png* (or any unique name of your choosing).
3. Copy the desired .png image to the folder above.
4. Rename it *Banner.png*.
5. Return to the Management Portal and refresh your browser (CTRL+F5 or F12) to display the changes.

Note: The image must be a .png format. Using any other format will cause an error.

Tip: The default Keyfactor logo size is 144 x 47 pixels. If you choose a different sized image, the spacing on the browser screens will change.

2.2.1.3 Keyfactor Command Appsetting.json Files

Select configuration settings for Keyfactor Command can be found in appsetting.json files within each area subdirectory of your Keyfactor Command installation (e.g. /WebConsole/Configuration/appsettings.json). There are settings that are unique to just one file (for example, connection settings for the orchestrator API) and settings that can be find in all of the files (for example, SQL retry settings). When a setting is found across all files, it should typically be configured identically across all files. For more information, see:

- Keyfactor Command WebConsole Services on page 786
- Keyfactor Command Claims Proxy Services on page 775
- Keyfactor Command Web Agent Services below
- Keyfactor Command Keyfactor API on page 777
- Keyfactor Command Service Job Settings on page 778
- Keyfactor Command Changing SQL Retry Settings on page 787

Keyfactor Command Web Agent Services

The WebAgentServices appsettings.json configuration file allows you to change default orchestrator API installation configuration settings.

To update the appsettings.json file:

1. Navigate to the WebAgentServices/Configuration folder on your server (located by default at: *C:\Program Files /Keyfactor/Keyfactor Platform/*).
2. Browse to open the *appsettings.json* file in a text editor (e.g. Notepad) and adjust the values as needed.
3. Save the files.

Table 77: Appsetting.json File Parameters - WebAgentServices

Setting	Description
ActiveDirectoryEnforced	This should be set to false if you are not using Active Directory. An IIS reset will be required to apply this setting if you change it.
CheckAuthCertificateRevocationStatus	Enter *true* to check the revocation status of the certificate provided for client certificate authorization, if being used, otherwise, enter *false*. The default is *true*.
ExtensionsDirectory	Enter the file path to the extensions to be loaded by the extension loader (for registration handler, workflow step, etc… support). The default value is *Extensions* (translates to: *C:/Program Files/Keyfactor/Keyfactor Platform/WebAgentServices/Extensions*)
MaxRequestSizeKb	This application setting is only used for the Orchestrator CA Sync controller. It is used to configure the CA sync batch size. We used to just grab the maximum request size from the IIS configuration settings, but that can no longer be done in .NET 6. The default is 5000.
NLogConfigFile	Enter the file path to the Nlog_Orchestrators.config file. The default is *C:/Program Files\Keyfactor/Keyfactor Platform/WebAgentServices/Configuration*

Setting	Description
SqlRetryConfiguration	SQL retry settings (seeKeyfactor Command Changing SQL Retry Settings on page 787 for more information). **Setting** / **Description** NumberOfTries — The number of times it will try the connection before an exception is thrown DeltaTime — The preferred gap time (in seconds) to delay before retry MaxTimeInterval — The maximum gap time (in seconds) for each delay time before retry
IdpInitializationTimeoutSeconds:	Duration of timeout (in seconds) used to make sure all authentication schemes are available before we allow users to login.

```
{
  "ActiveDirectoryEnforced": true,
  "CheckAuthCertificateRevocationStatus": "true",
  "ExtensionsDirectory": "Extensions",
  "MaxRequestSizeKb": "5000",
  "NLogConfigFile": "Configuration\\NLog_Orchestrators.config",
  "IdpInitializationTimeoutSeconds": 10
  "SqlRetryConfiguration": {
    "NumberOfTries": "5",
    "DeltaTime": "00:00:00.5",
    "MaxTimeInterval": "00:02:00"
  }
}
```

Figure 408: Sample WebAgentServices Appsettings.json File

Keyfactor Command Claims Proxy Services

The new ClaimsProxy appsettings.json configuration file allows you to view or modify the configuration settings for the Claims Proxy service.

To update the appsettings.json file:

1. Navigate to the ClaimsProxy/Configuration folder on your server (located by default at: *CC:\Program Files\Keyfactor\Keyfactor Platform\ClaimsProxy\Configuration*).
2. Browse to open the *appsettings.json* file in Notepad++ and adjust the values as needed as per Table 78: Appsetting.json File Parameters - ClaimsProxy.
3. Save the file.

Table 78: Appsetting.json File Parameters - ClaimsProxy

Setting	Description
NLogConfigFile	Enter the file path to the Nlog_Orchestrators.config file. The default is *C:/Program Files\Keyfactor/Keyfactor Platform/WebAgentServices/Configuration*
ExtensionsDirectory	Enter the file path to the extensions to be loaded by the extension loader (for registration handler, workflow step, etc… support). The default value is *Extensions* (translates to: *C:/Program Files/Keyfactor/Keyfactor Platform/WebAgentServices/Extensions*)
SqlRetryConfiguration	SQL retry settings (seeKeyfactor Command Changing SQL Retry Settings on page 787 for more information).

Setting	Description
NumberOfTries	The number of times it will try the connection before an exception is thrown
DeltaTime	The preferred gap time (in seconds) to delay before retry
MaxTimeInterval	The maximum gap time (in seconds) for each delay time before retry

ReverseProxy	This setting should only be changed on advice and guidance from Keyfactor Support.
Clusters	Clusters address field is populated by the URL for the API that is entered on the config wizard orchestrators tab and is automatically added to this file when the config wizard is saved.

```
"NLogConfigFile": "Configuration\\NLog_ClaimsProxy.config",
"ExtensionsDirectory": "Extensions",
"SqlRetryConfiguration": {
  "NumberOfTries": "5",
  "DeltaTime": "00:00:00.5",
  "MaxTimeInterval": "00:02:00"
},
"ReverseProxy": {
  "Routes": {
    "catch-all-ingress": {
      "ClusterId": "api-cluster",
      "AuthorizationPolicy": "RequireAuthenticatedUserPolicy",
      "Match": {
        "Path": "{**catch-all}"
      },
      "AllowAnonymous": false
    }
  },
  "Clusters": {
    "api-cluster": {
      "Destinations": {
        "api-node-1": {
          "Address": "https://keyfactor243.keyexample.com/KeyfactorAPI/"
        }
      }
    }
  }
}
```

Figure 409: Sample ClaimsProxy Appsettings.json File

Keyfactor Command Keyfactor API

The KeyfactorAPI appsettings.json configuration file allows you to view or change the Keyfactor Command Keyfactor API installation configuration settings.

To update the appsettings.json file:

1. Navigate to the KeyfactorAPI/Configuration folder on your server (located by default at: *C:\Program Files\Keyfactor\Keyfactor Platform\KeyfactorAPI\Configuration*).
2. Browse to open the *appsettings.json* file in a text editor (e.g. Notepad) and adjust the values as needed.
3. Save the files.

Table 79: Appsetting.json File Parameters - KeyfactorAPI

Setting	Description
NLogConfigFile	Enter the file path to the Nlog_KeyfactorAPI.config file. The default is *C:/Program Files\Keyfactor/Keyfactor Platform/KeyfactorAPI\Configuration)*
ExtensionsDirectory	Enter the file path to the extensions to be loaded by the extension loader (for registration handler, workflow step, etc… support). The default value is *Extensions* (translates to: *C:/Program Files/Keyfactor/Keyfactor Platform/KeyfactorAPI/Extensions*)
ActiveDirectoryEnforced	This should be set to false if you are not using Active Directory. An IIS reset will be required to apply this setting if you change it.
SqlRetryConfiguration	SQL retry settings (seeKeyfactor Command Changing SQL Retry Settings on page 787 for more information).

Setting	Description
NumberOfTries	The number of times it will try the connection before an exception is thrown
DeltaTime	The preferred gap time (in seconds) to delay before retry
MaxTimeInterval	The maximum gap time (in seconds) for each delay time before retry

```
{
  "NLogConfigFile": "Configuration\\NLog_KeyfactorAPI.config",
  "ExtensionsDirectory": "Extensions",
  "ActiveDirectoryEnforced": true,
  "SqlRetryConfiguration": {
    "NumberOfTries": "5",
    "DeltaTime": "00:00:00.5",
    "MaxTimeInterval": "00:02:00"
  }
```

Figure 410: Sample KeyfactorAPI Appsettings.json File

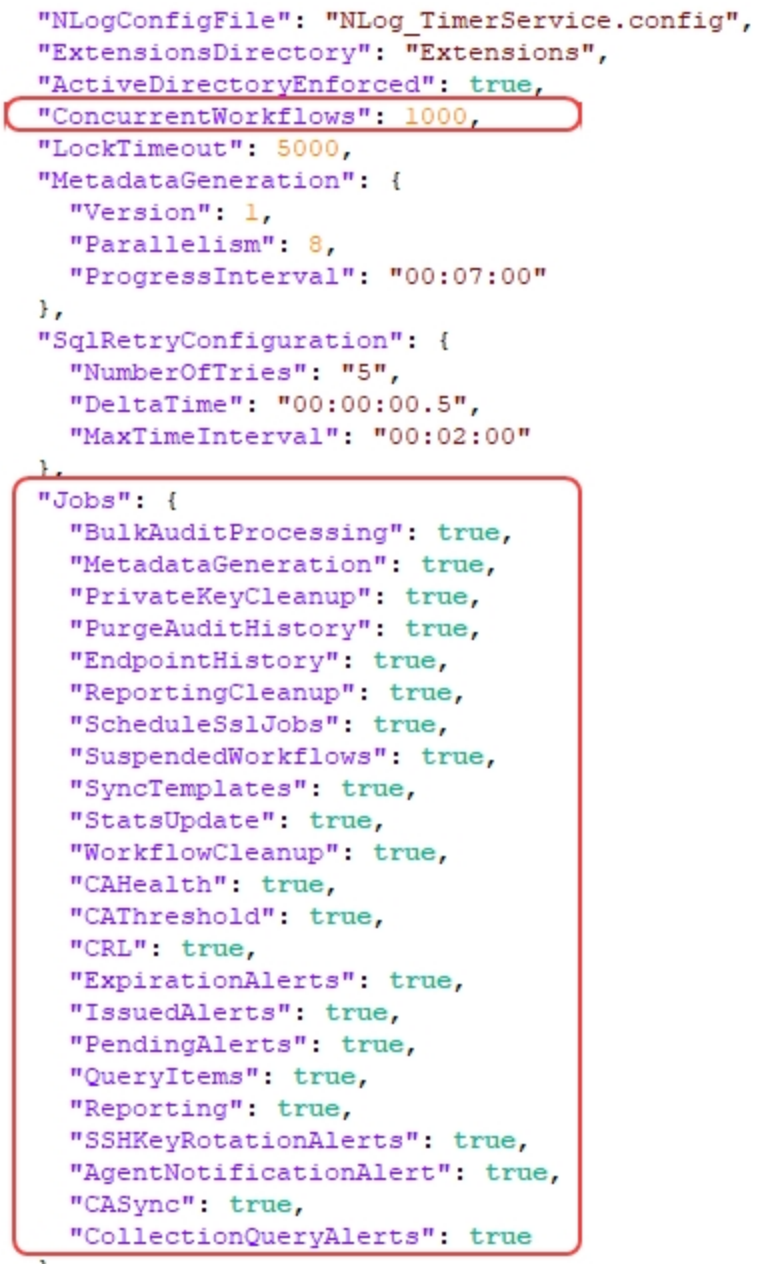
Keyfactor Command Service Job Settings

The Service appsettings.json file allows you to view or change the Keyfactor Command Service installation and configuration settings. By default, the Keyfactor Command Service job sets all service jobs to run based on the configuration wizard setting (see Service Tab on page 2801). The setting for select service jobs can be changed in the appsettings.json file.

To update the appsettings.json file for service configuration:

1. Navigate to the Service/Configuration folder on your server (located by default at: *C:\Program Files\Keyfactor\Keyfactor Platform\Service\Configuration*).

2. Browse to open the *appsettings.json* file in a text editor (e.g. Notepad) and adjust the values as needed as per Table 81: Keyfactor Command Jobs Services and Table 80: Keyfactor Command Services Configuration Settings



```
"NLogConfigFile": "NLog_TimerService.config",
"ExtensionsDirectory": "Extensions",
"ActiveDirectoryEnforced": true,
"ConcurrentWorkflows": 1000,
"LockTimeout": 5000,
"MetadataGeneration": {
  "Version": 1,
  "Parallelism": 8,
  "ProgressInterval": "00:07:00"
},
"SqlRetryConfiguration": {
  "NumberOfTries": "5",
  "DeltaTime": "00:00:00.5",
  "MaxTimeInterval": "00:02:00"
},
"Jobs": {
  "BulkAuditProcessing": true,
  "MetadataGeneration": true,
  "PrivateKeyCleanup": true,
  "PurgeAuditHistory": true,
  "EndpointHistory": true,
  "ReportingCleanup": true,
  "ScheduleSslJobs": true,
  "SuspendedWorkflows": true,
  "SyncTemplates": true,
  "StatsUpdate": true,
  "WorkflowCleanup": true,
  "CAHealth": true,
  "CAThreshold": true,
  "CRL": true,
  "ExpirationAlerts": true,
  "IssuedAlerts": true,
  "PendingAlerts": true,
  "QueryItems": true,
  "Reporting": true,
  "SSHKeyRotationAlerts": true,
  "AgentNotificationAlert": true,
  "CASync": true,
  "CollectionQueryAlerts": true
```

Figure 411: Appsettings.json File for TimerService Settings

3. Save the file.

Table 80: Keyfactor Command Services Configuration Settings

<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td>NLogConfigFile</td><td>Enter the file path to the Nlog_KeyfactorAPI.config file. The default is C:/Program Files\Keyfactor/Keyfactor Platform/KeyfactorAPI\Configuration)</td></tr>
<tr><td>ExtensionsDirectory</td><td>Enter the file path to the extensions to be loaded by the extension loader (for registration handler, workflow step, etc… support). The default value is Extensions (translates to: C:/Program Files/Keyfactor/Keyfactor Platform/KeyfactorAPI/Extensions)</td></tr>
<tr><td>ActiveDirectoryEnforced</td><td>This should be set to false if you are not using Active Directory. An IIS reset will be required to apply this setting if you change it.</td></tr>
<tr><td>SqlRetryConfiguration</td><td>SQL retry settings (seeKeyfactor Command Changing SQL Retry Settings on page 787 for more information).
<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td>NumberOfTries</td><td>The number of times it will try the connection before an exception is thrown</td></tr>
<tr><td>DeltaTime</td><td>The preferred gap time (in seconds) to delay before retry</td></tr>
<tr><td>MaxTimeInterval</td><td>The maximum gap time (in seconds) for each delay time before retry</td></tr>
</table></td></tr>
<tr><td>MetadataGeneration</td><td>
<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td>Version</td><td>This timer service job iterates over the certificates in the database, looks fora value less than the version set here, and if true, sets the certs' property with the version number specified here.</td></tr>
<tr><td>Parallelism</td><td>The how many threads of the job run simultaneously.</td></tr>
<tr><td>ProgressInterval</td><td>How often (in milliseconds) the work from cache is saved to the database.</td></tr>
</table></td></tr>
<tr><td>LockTimeout</td><td>The wait time in milliseconds to acquire a lock on a job for High Availability.</td></tr>
</table>

Table 81: Keyfactor Command Jobs Services

Type	Service	Description	Notes
Maintenance	BulkAuditProcessing	Periodically add audit log entries for large jobs. Most audit log entries are added immediately at the time the activity generating the audit log takes place. However, some large jobs that might generate heavy server load (e.g. bulk revocation) save the audit log entries in a temporary location to reduce server load and then they are added to the audit log by this periodic job.	This job runs every 10 minutes.
Maintenance	MetadataGeneration	Periodically generate and assign metadata to certificates when they are imported into Keyfactor Command using a custom metadata extension.	This job runs every 15 minutes.
Maintenance	PrivateKeyCleanup	Periodically remove any stored private keys in the Keyfactor Command database that have expired and are eligible for deletion.	This job runs daily at 1:00 am. For more information about stored private keys, see Status Tab on page 23.
Maintenance	PurgeAuditHistory	Periodically remove any audit log history in the Keyfactor Command database that has expired and is eligible for deletion.	This job runs monthly on the first day of the month at 2:00 am. For more information, see Application Settings: Auditing Tab on page 608. Only audit logs belonging to unprotected categories are eligible for deletion.
Maintenance	EndpointHistory	Periodically remove any SSL endpoint history in the	This job runs daily at 1:00 am.

Type	Service	Description	Notes
		Keyfactor Command database that is eligible for deletion, based on the setting in Application Settings: Auditing Tab on page 608 (*SSL > Retain SSL Endpoint History (days)*).	
Maintenance	ReportingCleanup	Periodically remove records from temporary files generated while running reports.	This job runs daily at midnight.
Maintenance	ScheduleSSLJobs	Periodically identify and schedule SSL discovery and monitoring jobs.	This job runs every 5 minutes.
Other	SuspendedWorkflows	Periodically attempt to continue all suspended workflows that may be eligible to continue but have not done so due to locking conflicts. A locking conflict may occur if two users attempt to provide input to a workflow instance (e.g. approve a request) at exactly the same time.	This job runs daily at midnight.
Maintenance	SyncTemplates	Periodically synchronize certificate templates from the source (e.g. Active Directory) to pick up new templates.	This job runs every hour.
Maintenance	StatsUpdate	Periodically run the Microsoft SQL update statistics function in the Keyfactor Command database.	This job runs monthly on the first day of the month at 1:00 am.
Maintenance	WorkflowsCleanup	Periodically remove any completed workflow instances (both successful and failed) in the Keyfactor Command database that have aged X number of days	This job runs daily at midnight.

Type	Service	Description	Notes
		past the completion date (last modified date), where X is defined by the *Workflow Instance Cleanup Days* application setting (see Application Settings: Console Tab on page 602). The default value is 14 days.	
Alerts	CAHealth	Periodically send email alerts when a CA is not responding.	The schedule for this is user configurable (see Certificate Authority Monitoring on page 378).
Alerts	CAThreshold	Periodically send email alerts when a CA is issuing certificates or experiencing issuance failures outside of the established norms.	The schedule for this is user configurable (see Advanced Tab on page 365).
Alerts	CRL	Periodically send email alerts for certificate revocation lists (CRLs) that are approaching expiration.	The schedule for this is user configurable (see Adding or Modifying a Revocation Monitoring Location on page 211).
Alerts	ExpirationAlerts	Periodically send email alerts for certificates approaching expiration.	The schedules for these are user configurable. See Configuring an Expiration Alert Schedule on page 171.
Alerts	IssuedAlerts	Periodically send email alerts (typically to certificate requesters) for certificate requests made using a certificate template that requires manager approval that have been approved.	The schedule for this is user configurable (see Configuring an Issued Request Alert Schedule on page 193)..
Alerts	PendingAlerts	Periodically send email alerts (typically to certificate	The schedules for these are user config-

Type	Service	Description	Notes
		approvers) for certificate requests made using a certificate template that requires manager approval.	urable. See Configuring a Pending Request Alert Schedule on page 183.
Other	QueryItems	Periodically populates a cache of which certificates are in which collection in the database. This is used in for the workflow step types: *Certificate Entered Collection* and *Certificate Left Collection*	Runs every 10 minutes (see Workflow Definition Operations on page 235)
Other	Reporting	Deliver regularly scheduled reports via email or saved to a file system.	The schedules for these are user configurable (see Reports on page 91).
Alerts	SSHKeyRotationAlerts	Periodically send email notifications to SSH key users and/or administrators when a key is nearing the end of the key lifetime.	The schedule for this is user configurable (see Configuring a Key Rotation Alert Schedule on page 206).
Alerts	AgentNotificationAlert	Periodically runs a job that checks if an orchestrator has not checked in between job runs and sends an email notification as per settings in Application Settings: Agents Tab on page 614.	This is configurable at Application Settings: Agents Tab on page 614
Certificate Authority	CASync	Periodically synchronize certificates from certificate authorities.	The schedules for this are user configurable (see Certificate Authorities on page 349).
Alerts	CollectionQueryAlerts	Periodically update the temporary tables that store information on which certificates are in which certificate collections. These temporary tables (caches) are used to support faster	This value is user configurable with an application setting (see Application Settings: Console Tab on page 602). The default is 20 minutes.

Type	Service	Description	Notes
		processing of some systems.	
n/a	ConcurrentWorkflows	Sets the batch size used when suspended workflows are run by the Keyfactor Command service. Also used when running certificate entered collection and certificate left collection workflows to limit the number of certificates flowing through the workflow for each instance of the workflow initiated by the service.	The default value is 1000.

Keyfactor Command WebConsole Services

The WebConsole appsettings.json configuration file allows you to view or change the Keyfactor Command Management Portal installation configuration settings.

To update the appsettings.json file:

1. Navigate to the WebConsole/Configuration folder on your server (located by default at: *C:\Program Files /Keyfactor/Keyfactor Platform/Console/Configuration*).
2. Browse to open the *appsettings.json* file in a text editor (e.g. Notepad) and adjust the values as needed.
3. Save the files.

Table 82: Appsetting.json File Parameters - WebPortalServices

Setting	Description
NLogConfigFile	Enter the file path to the Nlog_Orchestrators.config file. The default is *C:/Program Files\Keyfactor/Keyfactor Platform/WebConsole/Configuration*
ExtensionsDirectory	Enter the file path to the extensions to be loaded by the extension loader (for registration handler, workflow step, etc… support). The default value is *Extensions* (translates to: *C:/Program Files/Keyfactor/Keyfactor Platform/WebConsole/Extensions*)
ActiveDirectoryEnforced	This should be set to false if you are not using Active Directory. An IIS reset will be required to apply this setting if you change it.

Setting	Description
SqlRetryConfiguration	SQL retry settings (see Keyfactor Command Changing SQL Retry Settings below for more information).

Setting	Description
NumberOfTries	The number of times it will try the connection before an exception is thrown
DeltaTime	The preferred gap time (in seconds) to delay before retry
MaxTimeInterval	The maximum gap time (in seconds) for each delay time before retry

```
"NLogConfigFile": "Configuration\\NLog_Portal.config",
"ExtensionsDirectory": "Extensions",
"ActiveDirectoryEnforced": true,
"SqlRetryConfiguration": {
  "NumberOfTries": "5",
  "DeltaTime": "00:00:00.5",
  "MaxTimeInterval": "00:02:00"
},
```

Figure 412: Sample WebConsole Services Appsettings.json File

Keyfactor Command Changing SQL Retry Settings

The settings for SQL retries are configured in the appsettings.json file in each of the Keyfactor Program folders: Service, WebConsole, ClaimsProxy, WebAgentServices, and KeyfactorAPI (located by default at: *C:\Program Files\Keyfactor\Keyfactor Platform/...*). Any changes must be made to match in each appsettings.json file.

To update the appsettings.json file for SQL configuration:

1. Navigate to the /Configuration folder on your server for each of the Keyfactor Program folders.
2. Browse to open the *appsettings.json* file in a text editor (e.g. Notepad) and adjust the values as needed.
3. Save the files.

Table 83: Appsetting.json file - SQL Retry Parameters

Setting	Description
NumberOfTries	The number of times it will try the connection before an exception is thrown
DeltaTime	The preferred gap time (in seconds) to delay before retry
MaxTimeInterval	The maximum gap time (in seconds) for each delay time before retry

```
"NLogConfigFile": "Configuration\\NLog_KeyfactorAPI.config",
"ExtensionsDirectory": "Extensions",
"ActiveDirectoryEnforced": true,
"SqlRetryConfiguration": {
  "NumberOfTries": "5",
  "DeltaTime": "00:00:00.5",
  "MaxTimeInterval": "00:02:00"
},
```

Figure 413: Sample Appsettings.json File for SQL Retry Settings

2.2.1.4 Keyfactor Command Service Executable

By default, the Keyfactor Command Service is run using the CMSTimerService.exe executable. This executable is found in the following directory, by default:

```
C:\Program Files\Keyfactor\Keyfactor Platform\Service
```

This executable file does not shipped in a signed state. In environments where this is a concern, it is possible to change the Keyfactor Command Service to instead run using the CMSTimerService.dll. This file does ship in a signed state.

To make this change:

1. On the Keyfactor Command server, open a command prompt (NOT a PowerShell window) using the “Run as Administrator” option and type the following (h:

   ```
   sc config KeyfactorTimerJobService binPath= "dotnet \"C:\Program
   Files\Keyfactor\Keyfactor Platform\Service\CMSTimerService.dll\""
   ```

   Note: If your Keyfactor Command is installed in a different location than the default, change this path to reflect your installation location.

2. Once the command completes, you should receive a success message:

   ```
   [SC] ChangeServiceConfig SUCCESS
   ```

3. To confirm that the Keyfactor Command Service will now run as the dll, you can open the Properties for the service in the services.msc and view the *Path to executable* field to confirm that it ends with the dll rather than the exe. Note that this field should also begin *dotnet* \ once you switch to the dll.

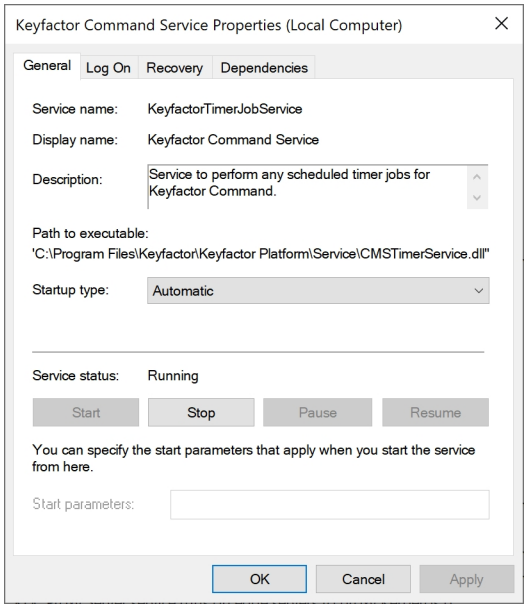


Figure 414: Switch the Keyfactor Command Service to Run as the CMSTimerService.dll

4. Restart the Keyfactor Command Service.

2.2.1.5 Configuring Key Recovery for Keyfactor Command

The following instructions for configuring CA-level key recovery within Keyfactor Command assume that your Microsoft CA is already configured for key recovery and that you have the key recovery agent (KRA) certificate available as a PFX file for import on the Keyfactor Command administration server. Instructions for configuring key recovery on a Microsoft CA are beyond the scope of this guide.

Tip: CA-level key recovery is supported for Microsoft CAs to allow recovery of private keys for certificates enrolled outside of Keyfactor Command. CA-level key archiving is not supported for enrollments done through Keyfactor Command. CA-level key recovery is not supported for EJBCA CAs. For enrollments done through Keyfactor Command for either Microsoft or EJBCA CAs, use Keyfactor Command private key retention (see Details Tab on page 387).

To configure your Keyfactor Command administration server to support key recovery:

1. Login on the Keyfactor Command administration server as the service account under which the Keyfactor Command application pool is running and open a command prompt. Alternately, if you

have previously logged on as this service account and created a user profile for the service account, you can open a command prompt as the service account using Shift-Right-Click and choose "Run as different user". Within the command prompt type the following to open the certificates MMC for the service account user:

```
certmgr.msc
```

2. Import the KRA PFX file into the service account user's personal certificate store.

This process needs to be repeated using the KRA certificate(s) from each CA for which you want to enable recovery within the Management Portal.

Note: To provide additional security over KRA private key(s), Keyfactor strongly recommends the use of a Hardware Security Module (HSM) such as the Thales NetHSM.

Tip: CA-level key recovery is not supported for EJBCA CAs. Instead, use private key retention within Keyfactor Command (see Details Tab on page 387).

2.2.2 SSH Reference

Please see the pages indicated for more detailed information about those specific SSH topics.

- SSH-Bash Orchestrator Job History Warning Resolution below
- SSH-SSSD Case Sensitivity Flag on the next page

2.2.2.1 SSH-Bash Orchestrator Job History Warning Resolution

Previously, it was unlikely the Bash orchestrator would fail during a sync job once it was configured correctly. With the introduction of SSSD support, there is additional validation the orchestrator must do as it applies the configured state that is being passed down from the server. Namely, we must validate that:

- The home directory known by SSSD falls directly underneath the LogonHomeDirectories setting value.
- The location of the authorized_keys directory as understood by SSHD is the home directory known by SSSD.
- The given logon must be resolvable in SSSD.

In the case where one or more of these criteria aren't valid assumptions, the logon won't be created or its keys will not be published. In this case, a message is returned on the Orchestrator Jobs page for the sync job with a Warning result (see Job History on page 516). These messages will continue to be returned until all issues are resolved. The intended resolution for this issue depends on the issue itself. See Table 84: Bash Orchestrator Job History Warning Resolution for examples of possible solutions to issues.

Table 84: Bash Orchestrator Job History Warning Resolution

Issue	Resolution
The home directory known by SSSD doesn't fall directly underneath the *LogonHomeDirectories* setting value.	Change the logon's home directory in the identity source that SSSD is pulling the identity from to be exactly one directory level under the configured value for the *LogonHomeDirectories* setting.
The location of the authorized_keys directory as understood by SSHD is not the home directory known by SSSD.	Modify the local SSHD configuration to ensure that the authorized_keys file can be resolved to the user's home directory and that the user's home directory is nested directly beneath the bash orchestrator's *LogonHomeDirectories* setting value.
A given logon cannot be resolved in SSSD.	• Ensure that the given logon name is valid in SSSD. **Tip:** The bash orchestrator will treat SSSD logon names as case sensitive despite the fact that the look up will succeed regardless of case sensitivity. Ensure that the logon name entered matches the logon name as presented by SSSD (see SSH-SSSD Case Sensitivity Flag below). • If the logon is found not be a valid logon on the server, delete the logon on the Keyfactor Command server and try adding the correct one.

2.2.2.2 SSH-SSSD Case Sensitivity Flag

As of RHEL 6 (SSSD package 1.6), a case_sensitive option was added to the valid list of parameters for a given provider in the */etc/sssd/sssd.conf* file. When this value is false, querying SSSD for a given user will return the username in all lower case, regardless of the casing in Active Directory. This value can be set to *Preserving*, which will return the casing used in the username in Active Directory.

The case sensitivity flag is important since attempting to create a new SSH logon in Keyfactor Command (see Adding Logons on page 585) requires that the username is entered as it appears in SSSD, regardless of this setting's value. Using *Preserving* makes the logons look like they do in Active Directory so it may be a less confusing experience for system administrators or those in charge of provisioning the accounts. If this flag is set to false, SSSD will return the name as all lower case characters to preserve POSIX compliance, which is how usernames will need to be entered into Keyfactor Command to create them.

Note: Besides the case-sensitive option setting, there are other SSSD settings that can affect how the username is presented which are not covered in this discussion.

Run the command below in your environment to determine how the username should be formatted.

```
getent passwd username@domain
```

Example:

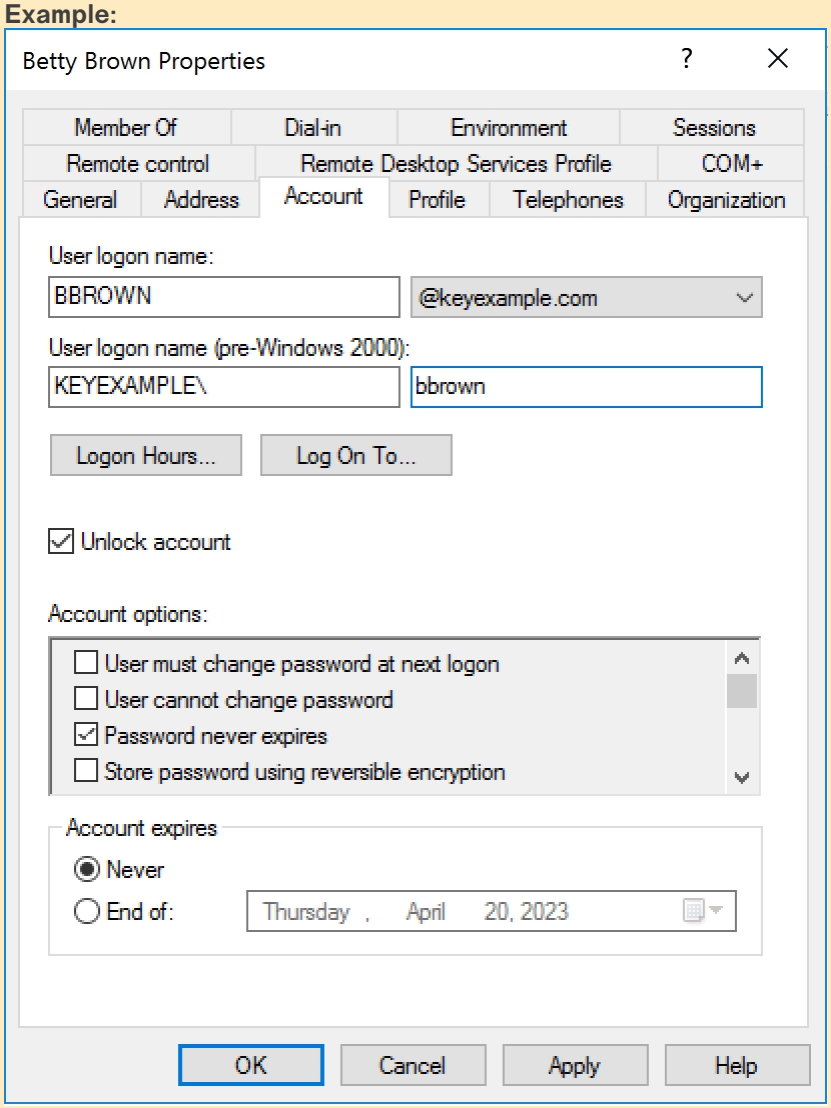


Figure 415: Active Directory Account Properties

The results for the above user with the setting as *false* would be: bbrown@keyexample.com.

```
getent passwd bbrown@keyexample.com
bbrown@keyexample.com:*1689201158:1689200513:Bbrown:/home/bbrown@keyexample.com:/bin/bash
```

The result for the above user with the setting as *Preserving* would be: BBROWN@keyexample.com.

```
getent passwd bbrown@keyexample.com
BBROWN@keyexample.com:*1689201158:1689200513:Bbrown:/home/BBROWN@keyexample.com:/bin/bash
```

Important: This value should not be changed once home directories have already been created on the server, even if done so prior to installing the Bash Orchestrator. Doing so will result in a conflict between Keyfactor Command's understanding of a login's casing and SSSD's. You will then receive an error until this logon is removed or its home directory is updated on the target server.

Example: User *BBROWN@keyexample.com* has a home directory */home/BBROWN@keyexample.com* that is out of compliance with SSSD known directory /home/*bbrown@keyexample.com*.The resolution of this error, in the case of the case_sensitivity property, is to either update the logon's home directory in AD or remove the logon's home directory on the local server and re-add it through Keyfactor Command.

Example: It is also possible for SSSD's understanding of a logon's home directory or account name to change if name of the domain changes in the SSSD config file. In this case, it's expected that the logon is removed from Keyfactor Command, in addition to its home directory on the Linux server, and re-added.

2.2.3 System Alerts

The System Alerts panel appears at the top of the Management Portal page just below the menu bar to display any errors or warnings found within Keyfactor Command. Click on the alerts indicator to toggle the System Alerts panel open/close. Warnings indicate things that may be of concern and appear in yellow. Errors indicate things that may be more urgent and appear in red. Click on the link included at the bottom of the system alert to be taken to the relevant page in the Management Portal to make the required configuration changes or corrections, if applicable. Some examples of conditions for which the system alerts appear include:

- The Keyfactor Command license is approaching expiration (warning)
- The Keyfactor Command license has expired (error)
- Certificate store job failures
- SSL scanning job failures
- NTLM authentication has been detected (and thus enrollment requests won't succeed)

Some system alerts are global and will appear on the system alerts panel regardless of where you are in the Management Portal. Other system alerts (such as some related to SSL scanning) are specific to a particular Management Portal page and will only appear when you are on that page.

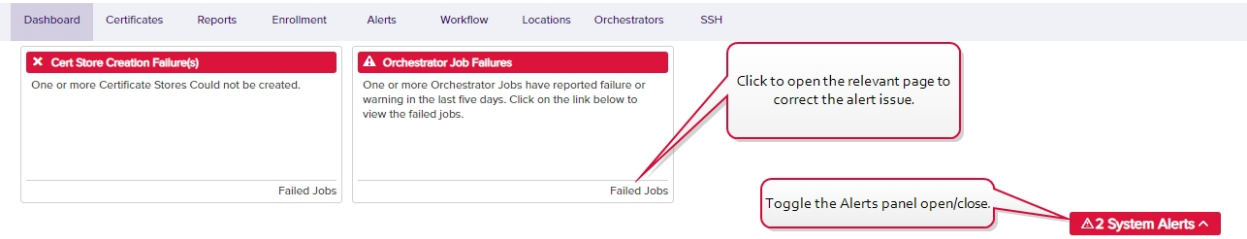


Figure 416: Management Portal Errors and Warnings

2.2.4 Log Monitoring

Logging information from your Keyfactor Command implementation is available in a variety of places:

- Dedicated text files for each application are written to the server.

  The logs for the various components of Keyfactor Command are saved, by default, under the local folder C:\Keyfactor\logs\.... For more information, see Editing NLog on page 796.

- The Windows event log on the Keyfactor Command server.

  For more information, see Keyfactor Command Windows Event IDs on page 806.

- The Audit Log in the Keyfactor Command Management Portal.

  Logs of auditable changes that affect your Keyfactor Command implementation—e.g. creation, change, or deletion of a record in an area of the product such as Certificates or Security—are viewable in the Management Portal, are output to a text file on the Keyfactor Command server, and can optionally be collected to a separate server for analysis with a centralized logging solution. For more information, see Audit Log on page 716.

In addition, transactions coming into the Keyfactor Command Management Portal are written into the IIS logs. For the most part, there is no need to look in the IIS logs unless you encounter a problem you need to troubleshoot. However, it is a good practice to monitor the text logs, the audit log, and the Windows event logs to make sure the system is operating smoothly and no errors are occurring.

By default, 10 main text logs are retained before the oldest ones are automatically deleted. Logs are rotated daily or when they reach a maximum file size, whichever comes first. Depending on the volume of log information you're generating, 10 logs may cover 10 days or a much shorter period. If you're using a centralized logging solution that runs daily to copy these to another location for analysis, the default log configuration of 10 logs with a maximum file size of 50 MB may be a sufficient retention policy. If you intend to analyze them in place on the Keyfactor Command server, you may wish to extend this retention setting.

In both the text-based logs and the Windows event logs, errors will generally appear with a tag of Error. For the text log, an error entry would look something like this, with more information following this line (and perhaps before it) with some further details:

```
2021-08-16 10:00:21.7105 CSS.CMS.CA.Client.CertificateAuthority [Error] - An error
occurred while reading the CA database.
```

Some errors may be transitory. For example, a CA synchronization may fail because a CA was down for maintenance and then succeed on the next try when the server is back up. If you find errors in your logs and need help tracking down their cause, contact Keyfactor support (support@key-factor.com).

When troubleshooting an error, it may be helpful to turn up the logging level in the Nlog.config file relevant to the component of interest to *debug* or *trace*. However, this can result in a large volume of messages that can be hard to wade through. It is sometimes useful to add further filters to the Nlog.config file relevant to the component of interest to filter out log traffic unrelated to the error you are trying to investigate. Some of the NLog files for the various log components contain pre-defined filters such as:

```
<when condition="ends-with('${logger}', 'WebSecurityContext') and level &lt;
LogLevel.Warn" action="Ignore" />
<when condition="ends-with('${logger}', 'AlertsController') and level &lt;
LogLevel.Warn" action="Ignore" />
<when condition="ends-with('${logger}', 'WebPrincipal') and level &lt; LogLevel.Warn"
action="Ignore" />
<when condition="ends-with('${logger}', 'CertStoreController') and level &lt;
LogLevel.Warn" action="Ignore" />
```

These filter out messages that contain the referenced string (e.g. WebSecurityContext) at the end of the log source string (e.g. CSS.CMS.Web.Security.WebSecurityContext) but only for messages that are at an Info, Debug or Trace level (less than Warn) as in this log line:

```
2021-08-11 04:38:04.0366 CSS.CMS.Web.Core.Security.WebSecurityContext [Trace] - User
'KEYEXAMPLE\ggant' (Cached) has area permission 'Reports_Read' as requested by 'Execute'
```

You can add more lines like this that do things like filter out the periodic report cleanup process, for example:

```
<when condition="ends-with('${logger}', 'ReportCleanupManager') and level &lt;
LogLevel.Warn" action="Ignore" />
```

You can also filter out messages based on all or part of the message. Say you want to look at CA synchronization messages, but want to eliminate some of the chatter related to that. You don't want to filter out all the CA synchronization source messages in that case, but you might choose to get rid of entries like this:

```
2020-05-20 08:41:00.0487 CSS.CMS.CA.Client.CertificateAuthorityConnector [Trace] - Fetch
succeeded
```

You can do that with a filter that looks like this:

```
<when condition="starts-with('${message}', 'Fetch succeeded') and level &lt;
LogLevel.Info" action="Ignore"/>
```

Note: For more information on how to make changes to your NLog configuration see Editing NLog below

Some informational, warning, and error messages generated by Keyfactor Command are coded in a manner to allow them to be redirected for output to the Windows Application event log. If you redirect these messages from being output to the event log to a file instead, they look something like:

```
2021-08-02 04:54:00.2260 CSS.CMS.Service.Jobs.CASync.LocalCASyncJob-EVENT [Info] -
eventID=200&categoryID=2&eventMessage=Beginning+Full+synchronization+of+Certificate+Auth
ority%3a+%27corpca02.keyexample.com
%5cCorpIssuing2.+Last+scan+time%3a+11%2f10%2f2020+10%3a20%3a00+AM%2c+last+row+read%3a+0%
27
```

The -EVENT tag (highlighted in red, above) is what codes these messages for redirection to the event log. There are two configuration lines in the NLog.config files for the various log components that relate to Windows event log redirection—the first formats the data correctly for event log usage and assigns a source to the messages while the second captures all the messages coded -EVENT, prevents them from going to the regular text log, and redirects them to the event log for all messages at info, warning or error level. Debug and trace level messages are not designed to be output to the event log. To reduce the volume of messages to the event log, you can change minlevel-l="Info" to minlevel="Warn" or minlevel="Error". Be aware that if you do this, more verbose messages (e.g. info level messages) will fall through to the text-based log.

```
    <target xsi:type="EventLog" name="eventLog" source="Keyfactor Command"
            eventId="${query-string:item=eventID}" category="${query-string:item=categoryID}" layout="${query-string:item=eventMessage}" />
</targets>
<rules>
  <!-- Internal ASP.NET logging, off by default -->
  <logger name="CSS.CMS.Web.API.SimpleTracer" minlevel="Off" writeTo="logfile" final="true" />
  <!-- Don't write events to the log file (log file should contain different, more verbose, logging) -->
  <logger name="CSS.CMS.Install.ConfigurationWizard.Console.Wizard" minlevel="Info" writeTo="console" />
  <logger name="*-EVENT" minlevel="Info" writeTo="eventLog" final="true" />
```

Figure 417: Nlog Configuration for Windows Event Logging

By default, messages redirected to the event log are marked with a source of *Keyfactor Command* for Keyfactor Command server, *Keyfactor Service* for the Keyfactor Command Service, and *Keyfactor Orchestrators* or *Keyfactor Orchestrator* for the Keyfactor Universal Orchestrator.

2.2.4.1 Editing NLog

Keyfactor Command provides extensive logging for visibility and troubleshooting. For more information about troubleshooting, see Troubleshooting on page 824.

By default, Keyfactor Command outputs logging information to both the Windows event log and the local file system, places file system log files in the C:\Keyfactor\logs directory, generates file system logs at the *Info* logging level, and stores the primary file system logs for two days before deleting them. If you wish to change these defaults you can open the configuration file for each type of log on each Keyfactor Command server where you wish to adjust logging, and edit the file in a text editor

(e.g. Notepad) using the “Run as administrator” option. Each Keyfactor component has its own NLog configuration file and NLog logging output path.

Note: By default, the filename for each component log is unique. This allows you to isolate and research issues on a component-by-component basis by viewing a specific log file. Alternatively, you may wish to change the default output filename to be the same for all logging components so all activity is reported in a single log file. You will note that the default Audit and Alert file names for each component (for those components that log audits or alerts) are the same so that all activity is logged in the same file across the platform for this reason.

Tip: If you use the default naming convention, and want to review an event that happened in the Management Portal, for instance, you would look in the Command_API_Log.txt and/or the Command_Portal_Log.txt.

Important: If you choose to name the log files the same across the platform, it is recommended that you also set the **maxArchiveFiles** values the same in each Nlog config file. If there is a different value for **maxArchiveFiles** for files with the same filename/location, the smallest value will override all others.

To make changes to your NLog configuration:

1. On each Keyfactor Command server where you wish to adjust logging, open a text editor (e.g. Notepad) using the “Run as administrator” option.

2. In the text editor, browse to open the desired Nlog.config file for the appropriate Keyfactor component (see Table 85: NLog.config Files for Keyfactor Command). The files are located in application subdirectories under the installed directory, which is the following by default:

   C:\Program Files\Keyfactor\Keyfactor Platform

3. Each Nlog.config file may have a slightly different layout than the portal example shown here, but it will contain many of the fields highlighted in the below figure. The fields you may wish to edit are:

   - The path and file name of the active Keyfactor Command log file:

     ```
     fileName="C:\Keyfactor\logs\Command_Portal_Log.txt"

     fileName="C:\Keyfactor\logs\Command_API_Log.txt"

     fileName="C:\Keyfactor\logs\Command_Service_Log.txt"

     fileName="C:\Keyfactor\logs\Command_OrchestratorsAPI_Log.txt"
     ```

```
fileName="C:\Keyfactor\logs\Command_Configuration_Log.txt"

fileName="C:\Keyfactor\logs\Command_ClaimsProxy_Log.txt"
```

Important: If you choose to change the path for storage of the log files, you will need to create the new directory (e.g. D:\KeyfactorLogs) and grant both the service account under which the Keyfactor Command Service is running and the service account under which the IIS application pool for Keyfactor Command is running full control permissions on this directory. These roles may be served by the same service account.

- The path and file name of previous days' log files:

  ```
  archiveFileName="C:\Keyfactor\logs\Command_Portal_Log_Archive_{#}.txt"

  archiveFileName="c:\Keyfactor\logs\Command_API_Log_Archive_{#}.txt"

  archiveFileName="C:\Keyfactor\logs\Command_Service_Log_Archive_{#}.txt"

  archiveFileName="C:\Keyfactor\logs\Command_OrchestratorsAPI_Log_Archive_
  {#}.txt"

  archiveFileName="C:\Keyfactor\logs\Command_Configuration_Log_Archive_{#}.txt"

  archiveFileName="c:\Keyfactor\logs\Command_ClaimsProxy_Log_Archive_{#}.txt"
  ```

  Keyfactor Command rotates log files daily and names the previous files using this naming convention.

- The path and file name of the active log file for alerting events:

  ```
  fileName="C:\Keyfactor\logs\Command_Alert_Log.txt"
  ```

  This entry is found on in the NLog.cong files on servers with the Keyfactor Command Service installed. Info level messages are written to this log whenever alerts (certificate expiration, pending certificate request, issued certificate, denied certificate request, or revocation monitoring) are run either as scheduled tasks or as tests. The log messages include the type of alert (e.g. expiration alert), the recipient of the alert (if an email was scheduled to be sent), and the alert subject line. You can change the level of logging in the log line that references writeTo="alertlogfile". These logs are generated separately from the general events to allow for separate tracking and log shipping of alerting events. By default, the alert log filename/location for all components is the same to allow for a central source for tracking alert events across the platform.

Note: This entry is not found in NLog_Configuration.config and NLog_ClaimsProxy.config files.

- The path and file name of previous days' Keyfactor Command log files for alert events:

```
archiveFileName="C:\Keyfactor\logs\Command_Alert_Log_Archive_{#}.txt"
```

Note: This entry is not found in NLog_Configuration.config and NLog_ClaimsProxy.config files.

- The path and file name of the active log file for auditable events:

```
fileName="C:\Keyfactor\logs\Command_Audit_Log.txt"
```

These logs are generated separately from the general events to allow for separate tracking of auditable events. By default, the audit log filename/location for all components is the same to allow for a central source for tracking auditable events across the platform.

Note: This entry is not found in NLog_ClaimsProxy.config files.

- The path and file name of previous days' Keyfactor Command log files for auditable events:

```
archiveFileName="C:\Keyfactor\logs\Command_Audit_Log_Archive_{#}.txt"
```

Note: This entry is not found in NLog_ClaimsProxy.config files.

- The level of log detail that should be generated for alert events and written to the alert logs:

```
name="EmailAlertLogger" minlevel="Info" writeTo="alertlogfile"
```

Note: This entry is not found in NLog_Configuration.config and NLog_ClaimsProxy.config files.

- The number of archive files to retain before deletion:

```
maxArchiveFiles="10"
```

This field is listed multiple times in most NLog.config files on a server with the Keyfactor Command Service installed—once for the main logging section, once for the alert logging section (if applicable), and once for the audit logging section. The default number of files to retain is 10 for the main log and the synchronization log but 14 for the alert log and audit

log. The alert and audit logs are small by default, and retaining a larger number of them shouldn't create a disk space issue.

- The maximum file size of each log file:

  ```
  archiveAboveSize="52428800"
  ```

  Once a log file reaches this size (50 MB by default), it will be rotated to archive even if the end of the day has not been reached.

- The level of log detail that should be generated:

  ```
  name="*" minlevel="Info" writeTo="logfile"
  ```

  This line applies to all the logs in the file. The default *Info* level logs error and some informational data but at a minimal level to avoid generating large log files. For troubleshooting, it may be desirable to set the logging level to *Debug* or *Trace*. Available log levels (in order of increasing verbosity) are:

  - OFF – No logging
  - FATAL – Log severe errors that cause early termination
  - ERROR – Log severe errors and other runtime errors or unexpected conditions that may not cause early termination
  - WARN – Log errors and use of deprecated APIs, poor use of APIs, *almost* errors, and other runtime situations that are undesirable or unexpected but not necessarily *wrong*
  - INFO – Log all of the above plus runtime events (startup/shutdown)
  - DEBUG – Log all of the above plus detailed information on the flow through the system
  - TRACE – Maximum log information—this option can generate VERY large log files

```
<!-- Keyfactor expects at least 2 targets to be present: 'logfile' and 'eventLog'. Please do not change the name attribute of these targets -->
<targets>
  <target xsi:type="AsyncWrapper" name ="LogfileAsync">
      <target xsi:type="File" name="logfile" fileName="C:\Keyfactor\logs\Command_API_Log.txt" layout="${longdate} ${uppercase:${activityId}} ${logger}
      [${level}] - ${message}"
          archiveFileName="c:\Keyfactor\logs\Command_API_Log_Archive_{#}.txt" archiveEvery="Day" archiveNumbering="Rolling" maxArchiveFiles="10"
          archiveAboveSize="52428800" KeepFileOpen="false" ConcurrentWrites="true"/>
  </target>
  <target xsi:type="File" name="alertlogfile" fileName="C:\Keyfactor\logs\Command_Alert_Log.txt" layout="${longdate} ${logger} [${level}] - ${message}"
          archiveFileName="c:\Keyfactor\logs\Command_Alert_Log_Archive_{#}.txt" archiveEvery="Day" archiveNumbering="Date" archiveDateFormat="yyyyMMdd"
          maxArchiveFiles="14" KeepFileOpen="false" ConcurrentWrites="true"/>
  <target xsi:type="File" name="auditlogfile" fileName="C:\Keyfactor\logs\Command_Audit_Log.txt" layout="${longdate} ${logger} [${level}] - ${message}"
          archiveFileName="c:\Keyfactor\logs\Command_Audit_Log_Archive_{#}.txt" archiveEvery="Day" archiveNumbering="Date" archiveDateFormat="yyyyMMdd"
          maxArchiveFiles="14" KeepFileOpen="false" ConcurrentWrites="true"/>
  <target xsi:type="OutputDebugString" name="String" layout="${longdate} ${logger}::${message}"/>
  <target xsi:type="Debugger" name="debugger" layout="${longdate} ${logger}::${message}"/>
  <target xsi:type="Console" name="console" layout="${logger} ${message}"/>
  <target xsi:type="EventLog" name="eventLog" source="Keyfactor Command"
          eventId="${event-properties:item=eventID}" category="${event-properties:item=categoryID}" layout="${event-properties:item=message}" />

</targets>
<rules>
  <!-- Internal ASP.NET logging, off by default -->
  <logger name="CSS.CMS.Web.API.SimpleTracer" minlevel="Off" writeTo="LogfileAsync" final="true" />
  <logger name="*-EVENT" minlevel="Info" writeTo="eventLog" final="true" />
  <logger name="*-AUDIT" minlevel="Info" writeTo="auditlogfile" final="true" />
  <logger name="EmailAlertLogger" minlevel="Info" writeTo="alertlogfile" final="true"/>
  <logger name="*" minlevel="Info" writeTo="LogfileAsync">
    <filters defaultAction="Log">
      <when condition="ends-with('${logger}', 'WebSecurityContext') and level &lt; LogLevel.Warn" action="Ignore" />
      <when condition="ends-with('${logger}', 'AlertsController') and level &lt; LogLevel.Warn" action="Ignore" />
      <when condition="ends-with('${logger}', 'CertStoreController') and level &lt; LogLevel.Warn" action="Ignore" />
    </filters>
  </logger>
</rules>
```

Figure 418: Nlog_KeyfactorAPI.config

4. Change the respective files and save your changes.

Tip: Some of the NLog.config files also contain a few filters that eliminate log messages of certain types from logging at debug and trace level as they can be chatty and may not be helpful except in special cases. You may find it helpful to emulate these with filters of your own when troubleshooting. For example, the following filters are found by default in the Keyfactor API and Management Portal log configurations:

```
<filters defaultAction="Log">
   <when condition="ends-with('${logger}', 'WebSecurityContext') and level < LogLevel.Warn"
action="Ignore" />
   <when condition="ends-with('${logger}', 'AlertsController') and level < LogLevel.Warn"
action="Ignore" />
   <when condition="ends-with('${logger}', 'CertStoreController') and level < LogLevel.Warn"
action="Ignore" />
   <when condition="starts-with('${logger}', 'Microsoft.') and level < LogLevel.Warn" action-
n="Ignore" />
</filters>
```

You can add additional filters to remove further messages. For example, this configuration shows logs related to reading certificate templates filtered out:

```
<filters defaultAction="Log">
   <when condition="ends-with('${logger}', 'WebSecurityContext') and level < LogLevel.Warn"
action="Ignore" />
```

```
   <when condition="ends-with('${logger}', 'AlertsController') and level < LogLevel.Warn"
action="Ignore" />
   <when condition="ends-with('${logger}', 'CertStoreController') and level < LogLevel.Warn"
action="Ignore" />
   <when condition="starts-with('${logger}', 'Microsoft.') and level < LogLevel.Warn" action-
n="Ignore" />
   <when condition="ends-with('${logger}', 'TemplateSecurityReader') and level <
LogLevel.Error" action="Ignore" />
   <when condition="ends-with('${logger}', 'CertificateTemplateReader') and level <
LogLevel.Error" action="Ignore" />
</filters>
```

The first, for TemplateSecurityReader messages, will filter out log messages such as these:

```
2023-10-11 03:59:15.0183  Keyfactor.ActiveDirectory.PKI.TemplateSecurityReader [Debug] -
Template: CorpWebServer
2023-10-11 03:59:15.0183  Keyfactor.ActiveDirectory.PKI.TemplateSecurityReader [Trace] -
KEYEXAMPLE\jsmith (S-1-5-21-2996571167-427620427-431932065-1000) mentioned for Enroll
2023-10-11 03:59:15.0183  Keyfactor.ActiveDirectory.PKI.TemplateSecurityReader [Trace] -
Access control type is 'Allow'
```

The second, for CertificateTemplateReader, will filter out log messages such as these:

```
2023-10-11 03:59:22.6312  Keyfactor.ActiveDirectory.PKI.CertificateTemplateReader [Debug] -
Leaving 'Load' method with 475 certificate template objects from AD
2023-10-11 03:59:22.6312  Keyfactor.ActiveDirectory.PKI.CertificateTemplateReader [Debug] -
Creating CertificateTemplate objects
2023-10-11 03:59:22.6312  Keyfactor.ActiveDirectory.PKI.CertificateTemplateReader [Trace] -
Adding Certificate Template 'User'.
```

Both these types of messages relate to building the cache of certificate authorities, templates and permissions that Keyfactor Command uses for enrollment. These messages are specific to Microsoft templates; other types of messages are generated for EJBCA templates. The cache is rebuilt every 10 minutes, so these messages will appear every 10 minutes. If you have a large number of templates and/or users granted rights to templates in your environment, the messages can be overwhelming if you enable debug or trace logging and you don't happen to be looking for template information.

Table 85: NLog.config Files for Keyfactor Command

Configuration File	Description
\WebConsole\Configuration\NLog_Portal.-config	The Portal file is for logging any activity to do with the Keyfactor CommandManagement Portal, including users connecting to the portal, loading various pages in the portal, and taking actions. **Note:** Many actions taken in the Keyfactor CommandManagement Portal are carried out using the Keyfactor API and Keyfactor is migrating the product to use the Keyfactor API more and more, so this file will have less and less activity going forward.
\KeyfactorAPI\Configuration\NLog_ KeyfactorAPI.config	The KeyfactorAPI file is the primary file for logging activity related to making requests with the Keyfactor API. Since many of the functions in the Management Portal use the Keyfactor API, this log also includes activity related to running the Management Portal.
\Service\Configuration\NLog_Timer-Service.config	The Timer Service file logs activity related to scheduled and automated events within Keyfactor Command such as CA synchronization, scheduled alerts, and scheduled reports.
\WebAgentServices\Configuration\NLog_ Orchestrators.config	The Orchestrators, or OrchestratorsAPI, file logs activity related to Keyfactor Orchestrators API. Look here for messages related to orchestrators communicating with Keyfactor Command.
\Configuration\NLog_Configuration.config	The Configuration file logs activity related to running the Keyfactor Command configuration wizard only. It may be useful to increase the logging level on this one if you are experiencing installation or upgrade issues.
\ClaimsProxy\Configuration\NLog_Claim-sProxy.config	The ClaimsProxy file logs activity related to using an identity provider other than Active Directory for authentication. Look here for messages related to authenticating to Keyfactor Command using token authentication. **Note:** The ClaimsProxy log offers a further option for extended logging beyond that found for the other logs since the type of messages generated for claims-related issues tend to occur not entirely within the realm of Keyfactor Command. To

Configuration File	Description
	configure extended logging, look in the targets section of the file for the target of type File: `<targets>` `  <target xsi:type="AsyncWrapper" name ="Logfile Async">` ` <target xsi:type="File" name="logfile" fileName="C:\Keyfactor\logs\Command_ClaimsProxy_Log.txt" layout="${longdate} ${logger} [${level}] - ${message}" archiveFileName-e="c:\Keyfactor\logs\Command_ClaimsProxy_Log_Archive_{#}.txt" archiveEvery="Day" archiveNumbering="Rolling" maxArchiveFiles-s="10" archiveAboveSize="52428800" KeepFileOpen="false" ConcurrentWrites="true"/>` ` </target>` ` <target xsi:type="OutputDebugString" name-e="String" layout="${longdate} ${logger}::${message}"/>` ` <target xsi:type="Debugger" name="debugger" layout="${longdate} ${logger}::${message}"/>` `</targets>`  Within this target, look for the reference to ${message} and modify the message structure to ${message}${newline}${microsoftconsolelayout} like so:  `<targets>` ` <target xsi:type="AsyncWrapper" name ="Logfile Async">` ` <target xsi:type="File" name="logfile" fileName="C:\Keyfactor\logs\Command_ClaimsProxy_Log.txt" layout="${longdate} ${logger} [${level}] - ${message}${newline}${microsoftconsolelayout}" archiveFileName-e="c:\Keyfactor\logs\Command_ClaimsProxy_Log_Archive_{#}.txt" archiveEvery="Day" archiveNumbering="Rolling" maxArchiveFiles-s="10" archiveAboveSize="52428800" KeepFileOpen="false" ConcurrentWrites="true"/>` ` </target>` ` <target xsi:type="OutputDebugString" name-e="String" layout="${longdate} ${logger}::${message}"/>`

Configuration File	Description

	
	`<target xsi:type="Debugger" name-e="debugger" layout="${longdate} ${log-ger}::${message}"/>`
	`</targets>`
	
	Save the file. There is no need to perform an iisreset.
	
	Extended messages are affected by the log level setting (e.g. Info, Debug, Trace), so set this value to Debug or Trace as well for full logging.

2.2.4.2 Audit Log Output to a Centralized Logging Solution

Keyfactor Command audit logging supports collecting audit entries in real time, as they are generated, to a separate server for analysis by a centralized logging solution. A variety of solutions can be supported. Typically the logs are either delivered to an rsyslog daemon on a Linux server, where they are consolidated with other logs and delivered on to a centralized solution, or delivered straight into the receiving pipeline of a centralized solution using a tool such as Splunk or Logstash. Delivery of the logs over a TLS connection is supported for backend solutions that support this option. Configuration of a centralized logging solution for delivery of the audit logs to a backend solution is beyond the scope of this guide. However, a sample rsyslog.conf file showing typical TLS configuration can be found in Prepare for External Log Shipping over TLS (Optional) on page 2775 in the *Keyfactor Command Server Installation Guide*.

The log output settings can be initially configured during installation and can be updated on the auditing tab of the applications settings page. The application settings that relate to log output are:

- Host Name
  Set this to the fully qualified domain name of the server that will be receiving the logs.
- Port
  Set this to the TCP port on which your log receipt application is listening to receive the logs. The default value is 514 (the default rsyslog port).
- Use SysLog Server
  This option defaults to **False**. Set it to **True** to enable delivery of logs to an outside server.
- Use TLS Connection
  This option defaults to **False**. Set it to **True** to enable delivery of logs to an outside server over TLS.

When you click **Save**, Keyfactor Command will verify that a connection can be made to the specified server on the specified port.

2.2.4.3 Keyfactor Command Windows Event IDs

Both Keyfactor Command and Keyfactor Orchestrators generate Windows event log messages for both normal activity and errors in the Windows application event log. Table 86: Keyfactor Command Windows Event IDs shows some of the more common event IDs generated by the Keyfactor Command server (source Certificate Management System or CMS Timer Job Servce). Table 88: Keyfactor Universal Orchestrator Windows Event IDs shows some of the more common event IDs generated by the Keyfactor Orchestrator (source Certificate Management System Agent). Depending on the features in use on your server, you may not see all these events in your log. These codes can be useful to set up log analysis platforms such as Splunk and Kibana.

Table 86: Keyfactor Command Windows Event IDs

Event ID	Task Category	Description
200	CA Synchron-ization	Incremental CA synchronization started
201	CA Synchron-ization	Incremental CA synchronization finished
210	CA Synchron-ization	An error occurred during CA synchronization
220	CA Synchron-ization	Unable to connect to the CA during incremental CA synchronization
221	CA Synchron-ization	Unable to validate Keyfactor Command product license
222	CA Synchron-ization	Unable to read the Keyfactor Command database during incremental CA synchronization
230	CA Synchron-ization	Unable to connect to the CA during full CA synchronization
300	Monitoring	Monitoring service started
301	Monitoring	Monitoring engine started
304	Monitoring	Monitoring service timer elapsed
305	Monitoring	Monitoring service execution skipped
306	Monitoring	Monitoring job completed successfully
307	Monitoring	Monitoring engine failed
310	Monitoring	Monitoring job completed with errors

Event ID	Task Category	Description
322	Monitoring	Unable to read the Keyfactor Command database during monitor job run
323	Monitoring	An error occurred refreshing a key rotation, cert expiration, CA Health, cert issued, pending cert, or query item alert service job
330	Monitoring	OCSP endpoint is unavailable
331	Monitoring	OCSP endpoint is responding successfully
340	Monitoring	An error occurred configuring an expiration alert
350	Monitoring	An error occurred configuring a pending alert
360	Monitoring	An error occurred configuring an SSL alert
370	Monitoring	An error occurred configuring the CRL
371	Monitoring	CRL endpoint location could not be contacted
372	Monitoring	CRL at the endpoint is stale (past the CA's next publish date for the CRL but not yet at the expiration date) **Note:** If a CRL is both in the warning period and stale, only the event log message for stale will appear in the log.
373	Monitoring	CRL at the endpoint is in the warning period configured for email alerts (X days before expiration)
374	Monitoring	CRL is in a good state
375	Monitoring	CRL at the endpoint has expired
380	Monitoring	An error occurred configuring a SSRS reporting job, CRL alert jobs, or certificate authority threshold jobs
390	Monitoring	Failed to configure the certificate authority threshold jobs
391	Monitoring	CA has failed to meet one of the threshold monitoring requirements
410	Web API	A general error occurred during a Keyfactor API request
411	Web API	Invalid token error occurred during a Keyfactor API request
413	Web API	Invalid template error occurred during a Keyfactor API request

Event ID	Task Category	Description
419	Web API	Invalid user error occurred during a Keyfactor API request
800	Timer Service	Keyfactor Command Service started
801	Timer Service	Keyfactor Command Service stopped
810	Maintenance	A general Keyfactor Command Service maintenance error occurred.
822	Timer Service	Unable to read the Keyfactor Command database during Keyfactor Command Service job
830	Timer Service	Keyfactor Command Service jobs failed to start (alerts, monitoring, sync, other)
930	Timer Service	An orchestrator job configuration failed
931	Timer Service	An orchestrator job execution failed
1001	Maintenance	Keyfactor Command product license is approaching expiration
1002	Maintenance	Audit logs failed to write to the audit log destination
1900	Configuration Wizard	The configuration wizard was started
1910	Configuration Wizard	The configuration wizard finished
1911	Configuration Wizard	The configuration wizard database creation process started
1912	Configuration Wizard	The configuration wizard database upgrade process started
1913	Configuration Wizard	The configuration wizard database conversion process started
1914	Configuration Wizard	The configuration wizard database upgrade process completed successfully
1915	Configuration Wizard	The configuration wizard database creation process completed successfully
1916	Configuration Wizard	The configuration wizard database conversion process completed successfully
1920	Configuration	A general failure occurred for the configuration wizard

Event ID	Task Category	Description
	Wizard	
1921	Configuration Wizard	The configuration wizard database upgrade process failed
1922	Configuration Wizard	The configuration wizard database creation process failed
1940	Configuration Wizard	Configuration wizard general warning
1941	Configuration Wizard	Configuration wizard SSRS reporting config warning
1942	Configuration Wizard	Configuration wizard agent pool config warning
2000	Alert	Whitelist policy failure
2300	Expiration Renewal	Renewal handler was able to successfully renew a certificate
2310	Expiration Renewal	Renewal handler failed to renew a certificate
2800	User Authentic-ation	User login to Management Portal was authenticated
3000	Alert	Execution of an alert (pending, issued, expiration, or key rotation) configured in the Management Portal failed.
3001	Alert	Execution of an alert (pending, issued, expiration, or key rotation) configured in the Management Portal succeeded.
3002	Alert	Execution of an alert (pending, issued, expiration, or key rotation) configured in the Management Portal was canceled.
3003	Alert	Execution of an alert (pending, issued, expiration, or key rotation) configured in the Management Portal started.
3004	Alert	A CA threshold monitoring alert failed.
3005	Alert	A CA threshold monitoring alert succeeded.
3006	Alert	A CA threshold monitoring alert was canceled.
3007	Alert	A CA threshold monitoring alert started.
3008	Alert	A CRL alert for a revocation monitoring location configured in the

Event ID	Task Category	Description
		Management Portal failed.
3009	Alert	A CRL alert for a revocation monitoring location configured in the Management Portal succeeded.
3010	Alert	A CRL alert for a revocation monitoring location configured in the Management Portal was canceled.
3011	Alert	A CRL alert for a revocation monitoring location configured in the Management Portal started.
3012	Certificate Authority	Local CA sync failed.
3013	Certificate Authority	Local CA sync succeeded.
3014	Certificate Authority	Local CA sync was canceled.
3015	Certificate Authority	Local CA sync started.
3016	Other	Delivery of regularly scheduled reports has failed.
3017	Other	Delivery of regularly scheduled reports has succeeded.
3018	Other	Delivery of regularly scheduled reports has been canceled.
3019	Other	Delivery of regularly scheduled reports has started.
3020	Maintenance	The process to generate and assign metadata to certificates when they are imported into Keyfactor Command has started.
3021	Maintenance	The process to generate and assign metadata to certificates when they are imported into Keyfactor Command has failed.
3022	Maintenance	The process to generate and assign metadata to certificates when they are imported into Keyfactor Command has been canceled.
3023	Maintenance	The periodic process to generate and assign metadata to certificates when they are imported into Keyfactor Command has succeeded.
3024	Maintenance	The periodic process to remove any stored private keys in the Keyfactor Command database that have expired and are eligible for

Event ID	Task Category	Description
		deletion has started.
3025	Maintenance	The periodic process to remove any stored private keys in the Keyfactor Command database that have expired and are eligible for deletion has failed.
3026	Maintenance	The periodic process to remove any stored private keys in the Keyfactor Command database that have expired and are eligible for deletion has been canceled.
3027	Maintenance	The periodic process to remove any stored private keys in the Keyfactor Command database that have expired and are eligible for deletion has succeeded.
3028	Maintenance	The periodic process to add audit log entries for large jobs started.
3029	Maintenance	The periodic process to add audit log entries for large jobs failed.
3030	Maintenance	The periodic process to add audit log entries for large jobs was canceled.
3031	Maintenance	The periodic process to add audit log entries for large jobs succeeded.
3032	Maintenance	The periodic process to remove any audit log history in the Keyfactor Command database that has expired and is eligible for deletion started.
3033	Maintenance	The periodic process to remove any audit log history in the Keyfactor Command database that has expired and is eligible for deletion failed.
3034	Maintenance	The periodic process to remove any audit log history in the Keyfactor Command database that has expired and is eligible for deletion was canceled.
3035	Maintenance	The periodic process to remove any audit log history in the Keyfactor Command database that has expired and is eligible for deletion succeeded.
3036	Maintenance	The periodic process to remove any SSL endpoint history in the Keyfactor Command database that is eligible for deletion started.
3037	Maintenance	The periodic process to remove any SSL endpoint history in the Keyfactor Command database that is eligible for deletion failed.

Event ID	Task Category	Description
3038	Maintenance	The periodic process to remove any SSL endpoint history in the Keyfactor Command database that is eligible for deletion was canceled.
3039	Maintenance	The periodic process to remove any SSL endpoint history in the Keyfactor Command database that is eligible for deletion succeeded.
3040	Alert	The periodic process to update the temporary tables that store information on which certificates are in which certificate collections started.
3041	Alert	The periodic process to update the temporary tables that store information on which certificates are in which certificate collections failed.
3042	Alert	The periodic process to update the temporary tables that store information on which certificates are in which certificate collections was canceled.
3043	Alert	The periodic process to update the temporary tables that store information on which certificates are in which certificate collections succeeded.
3044	Maintenance	The periodic process to remove records from temporary files generated while running reports started.
3045	Maintenance	The periodic process to remove records from temporary files generated while running reports failed.
3046	Maintenance	The periodic process to remove records from temporary files generated while running reports was canceled.
3047	Maintenance	The periodic process to remove records from temporary files generated while running reports succeeded.
3048	Other	The periodic process to attempt to continue all suspended workflows that may be eligible to continue but have not done so due to locking conflicts started.
3049	Other	The periodic process to attempt to continue all suspended workflows that may be eligible to continue but have not done so due to locking conflicts failed.
3050	Other	The periodic process to attempt to continue all suspended workflows that may be eligible to continue but have not done so due to locking

Event ID	Task Category	Description
		conflicts was canceled.
3051	Other	The periodic process to attempt to continue all suspended workflows that may be eligible to continue but have not done so due to locking conflicts succeeded.
3052	Maintenance	The periodic process to identify and schedule SSL discovery and monitoring jobs started.
3053	Maintenance	The periodic process to identify and schedule SSL discovery and monitoring jobs failed.
3054	Maintenance	The periodic process to identify and schedule SSL discovery and monitoring jobs was canceled.
3055	Maintenance	The periodic process to identify and schedule SSL discovery and monitoring jobs succeeded.
3056	Maintenance	The periodic process to synchronize certificate templates from a source (e.g. Active Directory) to pick up new templates started.
3057	Maintenance	The periodic process to synchronize certificate templates from a source (e.g. Active Directory) to pick up new templates failed.
3058	Maintenance	The periodic process to synchronize certificate templates from a source (e.g. Active Directory) to pick up new templates was canceled.
3059	Maintenance	The periodic process to synchronize certificate templates from a source (e.g. Active Directory) to pick up new templates succeeded.
3060	Maintenance	The periodic process to run the Microsoft SQL update statistics function in the Keyfactor Command database started.
3061	Maintenance	The periodic process to run the Microsoft SQL update statistics function in the Keyfactor Command database failed.
3062	Maintenance	The periodic process to run the Microsoft SQL update statistics function in the Keyfactor Command database was canceled.
3063	Maintenance	The periodic process to run the Microsoft SQL update statistics function in the Keyfactor Command database succeeded.
3064	Maintenance	The periodic process to remove any completed workflow instances (both successful and failed) in the Keyfactor Command database that have aged past the date as defined in that application started.

Event ID	Task Category	Description
3065	Maintenance	The periodic process to remove any completed workflow instances (both successful and failed) in the Keyfactor Command database that have aged past the date as defined in that application failed.
3066	Maintenance	The periodic process to remove any completed workflow instances (both successful and failed) in the Keyfactor Command database that have aged past the date as defined in that application canceled.
3067	Maintenance	The periodic process to remove any completed workflow instances (both successful and failed) in the Keyfactor Command database that have aged past the date as defined in that application succeeded.
3068	Alert	An alert for a certificate collection workflow started.
3069	Alert	An alert for a certificate collection workflow failed.
3070	Alert	An alert for a certificate collection workflow canceled.
3071	Alert	An alert for a certificate collection workflow succeeded.
3072	Alert	An orchestrator alert that a notification alert started.
3073	Alert	An orchestrator alert that a notification alert failed.
3074	Alert	An orchestrator alert that a notification alert canceled.
3075	Alert	An orchestrator alert that a notification alert succeeded.
3076	Alert	An alert for a certificate collection workflow started.
9999		Unknown error

Table 87: Keyfactor Command Windows Event IDs for Audit Log

Value	Subcategory Name	Description
2001	Certificate	Certificate
2001	AuditingCertificateScheduledReplacement	Auditing Certificate Scheduled Replacement
2001	AuditingCertificateRequest	Certificate Request
2002	ApiApplication	API Application

Value	**Subcategory Name**	Description
2003	Template	Template
2004	CertificateQuery	Certificate Collection/Query
2005	ExpirationAlert	Expiration Alert
2005	ExpirationAlertDefinitionContextModel	Expiration Alert
2006	PendingAlert	Pending Alert
2006	PendingAlertDefinitionContextModel	Pending Alert
2007	ApplicationSetting	Application Setting
2008	IssuedAlert	Issued Alert
2008	IssuedAlertDefinitionContextModel	Issued Alert
2009	DeniedAlert	Denied Alert
2009	DeniedAlertDefinitionContextModel	Denied Alert
2010	ADIdentityModel	Security Identity
2011	SecurityRole	Security Role
2012	AuthorizationFailure	Authorization Failure
2013	CertificateSigningRequest	CSR
2014	ServerGroup	SSH Server Group
2015	Server	SSH Server
2016	DiscoveredKey	Rogue Key for Logon
2016	Key	SSH Key
2017	ServiceAccount	SSH Service Account
2018	Logon	SSH Logon
2019	SshUser	SSH User
2020	KeyRotationAlertDefinitionContextModel	SSH Key Rotation Alert
2021	CertificateStore	Certificate Store

Value	Subcategory Name	Description
2022	JobType	Orchestrator Job Type
2023	AgentSchedule	Orchestrator Job
2024	BulkAgentSchedule	Bulk Orchestrator Job
2025	CertificateStoreContainer	Store Container
2026	Agent	Orchestrator
2027	RevocationMonitoring	Monitoring
2028	License	License
2029	WorkflowDefinition	Workflow Definition
2030	WorkflowInstance	Workflow Instance
2031	WorkflowInstanceSignal	Workflow Instance Signal
2032	IdentityProvider	Identity Provider
2033	RoleClaimDefinition	Claim Definition
2034	PermissionSet	Permission Set

Table 88: Keyfactor Universal Orchestrator Windows Event IDs

Event ID	Task Category	Description
1500	SSL Discovery	Starting SSL discovery job
1510	SSL Discovery	Completed SSL discovery job
1520	SSL Discovery	Error while performing SSL discovery job
1600	SSL Monitor	Starting SSL monitoring job
1610	SSL Monitor	Completed SSL monitoring job
1620	SSL Monitor	Error while performing SSL monitoring job
1630	SSL Monitor	Error connecting to an endpoint during an SSL scan

Event ID	Task Category	Description
1640	SSL Monitor	Certificate approaching expiration found at endpoint during an SSL scan
2400	AnyAgent Inventory	Keyfactor Universal Orchestrator: Starting inventory job for an AnyAgent certificate store
2410	AnyAgent Inventory	Keyfactor Universal Orchestrator: Completed inventory job for an AnyAgent certificate
2420	AnyAgent Inventory	Keyfactor Universal Orchestrator: Error while performing inventory job for an AnyAgent certificate store
2500	AnyAgent Management	Keyfactor Universal Orchestrator: Starting management job for an AnyAgent certificate store
2510	AnyAgent Management	Keyfactor Universal Orchestrator: Completed management job for an AnyAgent certificate
2520	AnyAgent Management	Keyfactor Universal Orchestrator: Error while performing management job for an AnyAgent certificate store
2800	Audit Log	Keyfactor Universal Orchestrator: Starting fetch logs job
2810	Audit Log	Keyfactor Universal Orchestrator: Completed fetch logs job
2820	Audit Log	Keyfactor Universal Orchestrator: Error while performing fetch logs job
2900	Agent Service	Job manager for the Keyfactor Universal Orchestrator starting
2920	Agent Service	Job manager for the Keyfactor Universal Orchestrator stopped

2.2.5 License Expiration Monitoring and Rotation

As your license is approaching expiration, warnings will be written to the Windows event log on the server running the Keyfactor Command service 60 days, 30 days and 5 days in advance of the license expiration (or at the next start of the Keyfactor Command service that falls within these time periods) using event ID 1001.

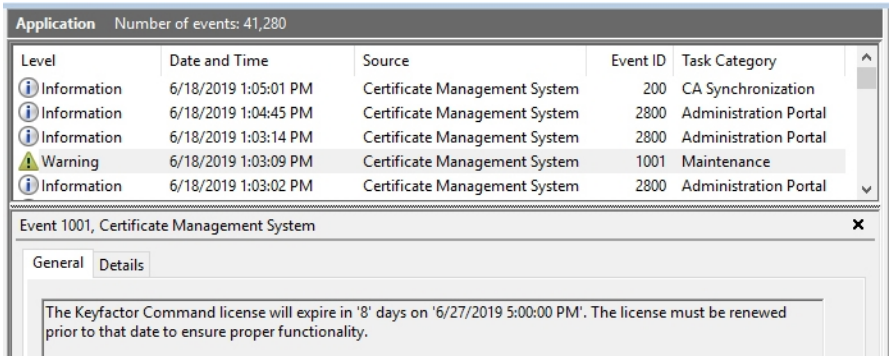


Figure 419: License Expiration Event Log

New primary Keyfactor Command licenses may be updated on the Licenses page of the Keyfactor Command Management Portal (see Licensing on page 768).

Tip: An error message of "Denied by Policy Module" with "Class is not licensed for use 0x80040112" on an attempt to enroll against a CA running the Keyfactor CA Policy Module can be an indication that the license for the policy module has expired.

New licenses for the Keyfactor CA Policy Module should be installed on the CA where the policy module is installed as follows:

1. On the CA where the policy module is installed, open the Certification Authority management tool.
2. In the Certification Authority management tool, right-click the CA name at the top of the tree and choose **Properties**.
3. In the Properties dialog for the CA on the CA Policy Module tab, confirm that the *Keyfactor Custom Policy Module* is the selected module and click **Properties**.
4. On the Licensing tab of the Policy Module Configuration Properties page, click **Upload License** and browse to locate the license file provided to you by Keyfactor. This file should have the extension CMSLICENSE.

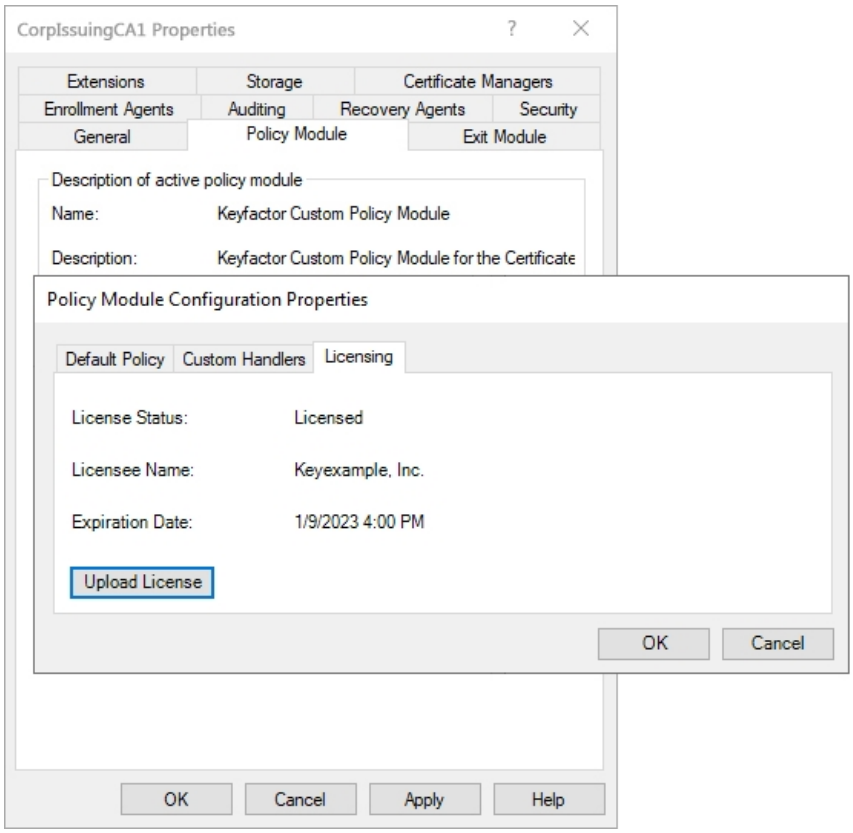


Figure 420: Upload a New Keyfactor Command License

5. Click **OK** as many times as needed to close the configuration dialogs and save the configuration.
6. Restart the CA services.

2.2.6 Disaster Recovery

Preparing for recovery of your Keyfactor Command server in the event of a disaster or in anticipation of a planned event such as a software upgrade or hardware migration involves backing up several different components. The bulk of the data for a Keyfactor Command server implementation is stored in a SQL database, so backing up this SQL database regularly is key. A portion of the data stored in this database is encrypted, so you will need the appropriate components to allow you to access this encrypted information.

Ideally, your disaster recovery plan would include backing up each server hosting a Keyfactor role as a whole entity. This greatly simplifies recovery. With a plan of this sort, you would need these backed up components:

- Keyfactor Servers

  Each server hosting a Keyfactor role—your Keyfactor Command servers, any Keyfactor orchestrators, etc.—should be backed up as entire entities with the full OS and installed applications.

- Your Keyfactor Command SQL Database

  All the Keyfactor Command data—both configuration data and synchronized data such as certificates—is contained within one database, which should be backed up regularly.

- The SQL server Database Master Key (DMK) and Service Master Key (SMK) for your SQL Database

  If you need to restore your SQL database to a different SQL server instance than the one from which it was backed up, you will need either the DMK or the SMK. There are pros and cons to restoring with each of these, so it can be useful to have both available when you make the restore decision. These only need to be backed up once unless you change either of these in SQL. See SQL Encryption Key Backup on the next page.

If backing up each server as a whole entity is not feasible or you would like to also back up components on the servers that differ from a stock install, consider including the following items for backup:

- Your nlog.config Files

  The various Keyfactor Command server components and most other Keyfactor products have an nlog.config file that sets the logging level for the product and the output path for the log files. If you have made any customizations to any of these configuration files, you may find it useful to make a backup of them. For Keyfactor Command server, the Nlog.config files for the various Keyfactor Command components are in application-specific subdirectories under the installation directory, which is by default:

  ```
  C:\Program Files\Keyfactor\Keyfactor Platform
  ```

  For example:

  ```
  C:\Program Files\Keyfactor\Keyfactor Platform\WebConsole\NLog_Portal.config
  C:\Program Files\Keyfactor\Keyfactor Platform\KeyfactorAPI\NLog_KeyfactorAPI.config
  ```

- Customizations for PAM Configuration

  If you have implemented a PAM solution and manually made configuration changes for this (see Installing Custom PAM Provider Extensions on page 743), you may want to include these files in a one-time backup.

- Event Handler Scripts

  If you have implemented any event handler scripts for alerting (see Using Event Handlers on page 218), you may want to backup these files.

- PowerShell Scripts for Workflow

  If you have implemented any custom PowerShell steps for that use external scripts (see Workflow Definitions on page 230), you may want to backup these files.

- Any Other Text-Based Files

  If you have modified any other text-based configuration files on your Keyfactor Command server (this is uncommon), you will want to have a one-time backup of these.

The process of restoring from backup depends on the components that have been affected. If only the Keyfactor Command server has been lost but the database is intact, the server may be restored

from backup and re-connected to the existing database. If a whole server backup does not exist, a fresh server may be installed, Keyfactor Command installed again and connected to the existing database, and any customized files restored or recreated. If the SQL database is lost, the database must be restored from backup along with either the DMK or SMK (see SQL Encryption Key Backup below).

For assistance with disaster recovery planning or implementation, please contact Keyfactor support (support@keyfactor.com).

2.2.6.1 SQL Encryption Key Backup

Keyfactor Command uses Microsoft SQL Server Encryption to encrypt portions of the database to protect secret data, including service account credentials. Understanding Keyfactor Command's use of SQL Server Encryption is important to a successful disaster recovery strategy.

SQL Server Encryption uses a SQL Server instance-level service master key (SMK) and a database-level database master key (DMK) to provide the top-level encryption hierarchy used when encrypting SQL data. The DMK is protected by one or more passwords and optionally the SMK. For an application—such as Keyfactor Command—to access SQL encrypted data, the application must either provide one of the DMK passwords or ask SQL Server to access the data via the SMK. Keyfactor Command uses the SMK. For more details on the mechanics of SQL Server Encryption and related disaster recovery procedures, see the SQL Server documentation.

When the Keyfactor Command database is created, the DMK is configured to be protected by the SMK and then the DMK password is set to a random value, which is not retained. This means the only way to get to the encrypted data is by leveraging the SMK, which happens automatically without any user interaction or the need to store the DMK password in a potentially insecure location.

Different restoration scenarios may require a backup of the SMK or the DMK or neither. Some restoration possibilities include:

- In the case where a Keyfactor Command database needs to be restored to the same SQL server where the backup was taken **and** the SQL Server software itself is not being restored, the correct SMK will still be present on the SQL server and restoration of the database itself is sufficient to be able to access the encrypted data.
- In the case where a Keyfactor Command database is being restored to a SQL Server with a different SMK (either a different SQL Server or the same SQL server that has been reinstalled or had its SMK changed), the encrypted data will be inaccessible because the server level SMK is not the same as it was when the DMK was created. In this scenario, either the DMK needs to be restored from the backup taken when the Keyfactor Command database was created or a known DMK password may be used to recover encrypted data within the Keyfactor Command database. To prepare for this scenario, the configuration wizard strongly encourages making a DMK backup when the Keyfactor Command database is created.
- In the case where a Keyfactor Command database needs to be restored to a SQL Server with a different SMK, the DMK cannot be restored and a DMK password is not known, a backup of the SMK may be used to restore the server, but this will affect any other databases on the server that make use of SQL encryption.

If no backup of the SMK or DMK exists, all DMK passwords are unknown, and the SQL server holding the SMK is lost, the encrypted data within Keyfactor Command is not recoverable (even with a database backup.)

To backup the DMK, as a user with *control* permission on the SQL server where the Keyfactor Command database is **select your Keyfactor Command database** and run the following SQL command:

```
BACKUP MASTER KEY TO FILE = 'path_to_file'
ENCRYPTION BY PASSWORD = 'SecurePassword#1234'
```

Replace ”path_to_file” with a path and filename for the output file. This can be either a local path on the SQL server or a UNC path. The selected output directory must be writable by the service account under which SQL Server is running. By default, the SQL backup directory has appropriate permissions. Replace ”SecurePassword#1234’ with a secure password to protect the file. Store the backup file and the password in a safe, well-documented location. For more information, see:

https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/back-up-a-database-master-key?view=sql-server-ver15
https://docs.microsoft.com/en-us/sql/t-sql/statements/backup-master-key-transact-sql?view=sql-server-2017

To backup the SMK, as a user with *control server* permission run the following SQL command on the SQL server where the Keyfactor Command database is:

```
BACKUP SERVICE MASTER KEY TO FILE = 'path_to_file'
ENCRYPTION BY PASSWORD = 'SecurePassword#1234'
```

Replace path_to_file with a path and filename for the output file. This can be either a local path on the SQL server or a UNC path. The selected output directory must be writable by the service account under which SQL Server is running. By default, the SQL backup directory has appropriate permissions. Replace SecurePassword#1234 with a secure password to protect the file. Store the backup file and the password in a safe, well-documented location. For more information, see:

https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/back-up-the-service-master-key?view=sql-server-ver15
https://docs.microsoft.com/en-us/sql/t-sql/statements/backup-service-master-key-transact-sql?view=sql-server-2017

To prepare for disaster recovery, you should have the DMK backup created during installation, an SMK backup, the passwords for these files and a recent database backup. You will likely only need either the DMK or the SMK if you need to restore to a SQL server instance other than the original SQL server instance, but it can be useful to have the flexibility to choose between the two at restoration time. If you need to restore to the original SQL server instance, you will only need a recent database backup and not either of the database keys. For information about restoring using the DMK or SMK, see Disaster Recovery on page 819.

2.2.7 SQL Database Migration

If you need to move your Keyfactor Command database from one SQL server to another, the process is similar to a controlled disaster recovery. You will need a backup of your Keyfactor Command database and the ability to decrypt the encrypted content within the database (see SQL Encryption Key Backup on page 821). By default, a new SQL server will have a different service master key (SMK) than your original SQL server. To support the migration, you have a few options:

- Set the SMK on the new server to match that of the old server and do a simple restore of the database. This may not be a feasible solution if there are any other applications on the new server that use SQL encryption.
- Temporarily add a known password to the database master key (DMK) on the Keyfactor Command database (if one is not known already).

To transfer a Keyfactor Command database between two SQL servers that do not share the same SMK, as a user with *control* permission on the Keyfactor Command database:

1. Add a known password to the DMK by issuing the following SQL command in the Keyfactor Command database. You can specify any password you want that meets the Windows password complexity rules.

   ```
   ALTER MASTER KEY ADD ENCRYPTION BY PASSWORD = 'SecurePassword#1234'
   ```

   Important: Note that at this point, in addition to the backup you are about to manually make, any automated backups of the Keyfactor Command database will contain this DMK password and anyone with access to the backup media and the password would be able to decrypt the sensitive information within the Keyfactor Command database.

2. Use your preferred SQL server tools to back up the database, copy the backup media to the target server, and restore the database on the target server.

3. Use the following SQL commands on the target server to manually open the DMK, protect the DMK with the target server's SMK, and remove the DMK password (referencing the password you used on your DMK):

   ```
   OPEN MASTER KEY DECRYPTION BY PASSWORD = 'SecurePassword#1234'

   ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY

   ALTER MASTER KEY DROP ENCRYPTION BY PASSWORD = 'SecurePassword#1234'
   ```

4. Open a new query window on the target server and use the following SQL to validate that the DMK is properly encrypted by the SMK and that the Keyfactor Command application will be able to ask SQL server to decrypt information in the database. The commands should run without error.

```
OPEN SYMMETRIC KEY [CMS_SecretsSymmetricKey] DECRYPTION BY CERTIFICATE [CMS_
SecretsCertificate];


CLOSE SYMMETRIC KEY [CMS_SecretsSymmetricKey]
```

5. On the source server, if you are not going to remove the Keyfactor Command database, issue the following SQL command to remove the DMK that was added (referencing the password you used on your DMK):

```
ALTER MASTER KEY DROP ENCRYPTION BY PASSWORD = 'SecurePassword#1234'
```

6. Delete the backup or securely store the backup media that was used, along with the temporary DMK password, as it can be used to obtain the encrypted Keyfactor Command information.

2.2.8 Troubleshooting

The following error conditions and general troubleshooting tips may be helpful in resolving issues with the Keyfactor Command server. Generally speaking, issues on installation or upgrade are often related to SQL connectivity or permissions. When Active Directory is used as the identity provider, certificate enrollment issues are often related to Kerberos configuration problems.

IIS Authentication Configuration

If authentication is not working as expected, it can be helpful to review whether the authentication configuration in IIS is correct for the identity provider you have configured. The correct IIS authentication configurations are:

- If you're using an identity provider other than Active Directory, the *KeyfactorAgents*, *KeyfactorAPI*, and *KeyfactorPortal* virtual directories should be set with Anonymous Authentication enabled and all other authentication methods disabled.

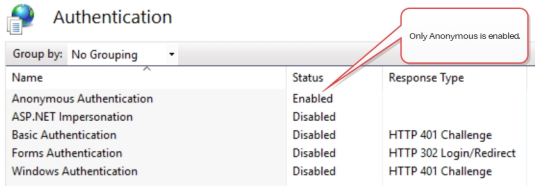


Figure 421: IIS Authentication for Virtual Directories with an Identity Provider Other Than Active Directory

- If you're using Active Directory, the default configuration for the *KeyfactorAgents*, *KeyfactorAPI*, and *KeyfactorPortal* virtual directories is Anonymous Authentication, Basic Authentication, and Windows Authentication enabled. You have the option to disable either Basic Authentication or Windows Authentication for the *KeyfactorPortal* and *KeyfactorAPI* virtual directories to force users to login with only one of these methods, if desired. The *KeyfactorAgents* virtual directory may be configured to Anonymous Authentication only if certificate-based authentication is in use for your orchestrators (see Appendices on page 3021).

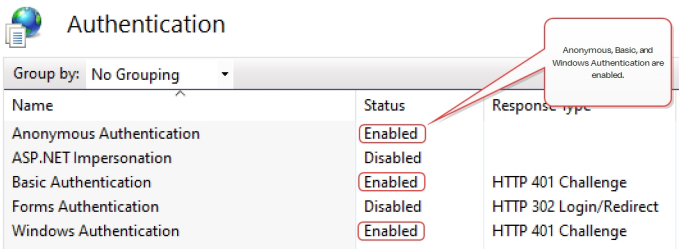


Figure 422: IIS Authentication for Virtual Directories with Active Directory

Note: If you make customizations to the authentication settings such as disabling Basic Authentication for the KeyfactorPortal virtual directory and subsequently enable the OAuth option in the configuration wizard, these customizations will be lost.

- The *KeyfactorAnalysis* and *KeyfactorProxy* virtual directories are set with only Anonymous Authentication for all identity provider configurations.
- At the site level (e.g. Default Web Site), only Anonymous Authentication should be enabled for all identity provider configurations.

Debug Logging and Error Messages

It is often helpful to enable debug logging on the server. For information on configuring this, see Editing NLog on page 796.

Once the logging is set at debug or trace level, it can be helpful to watch the logs live while activity is going on. There are tools on Windows with functionality similar to the Linux tail function to watch the log in real time. Notepad++, for example, has this functionality built in. Be sure to review all the logs that could be relevant. For example, installation and configuration messages are found in the configuration log. Messages related to using the Management Portal can be found in both the portal log and the Keyfactor API log.

Some messages in the Keyfactor API and orchestrators API logs include a correlation ID that helps to identify log messages that originated from the same request. The correlation ID is a randomly generated GUID that often appears just after the date in the log entry (C282ACA1-DED5-4F2E-B83B-F3F9E865E371 in the following example) and is the same for all log messages for the given request until the request completes.

```
2022-09-13 04:51:18.6884 C282ACA1-DED5-4F2E-B83B-F3F9E865E371 CSS.CMS.We-
b.KeyfactorApi.Controllers.Enrollment.Enrollment2Controller [Trace] - Starting PFX Enrollment Process
2022-09-13 04:51:19.0477 C282ACA1-DED5-4F2E-B83B-F3F9E865E371 Keyfactor.Com-
mand.Workflows.Engine.WorkflowGraph [Error] - Invalid O provided: Value must be Key Example, Inc or
Key Example.
```

General Errors

Below are some possible errors you might encountered and some suggested troubleshooting tips or solutions.

A connection was successfully established with the server, but then an error occurred during the login process. (provider: SSL Provider, error: 0 - The certificate chain was issued by an authority that is not trusted.)

You many encounter this error when trying to install or upgrade to Keyfactor Command version 10 or later:

```
2022-03-04 09:58:55.7262 CSS.CMS.Install.ConfigurationWizard.ViewModels.DatabaseViewModel [Error] -
Unable to configure database
2022-03-04 09:58:55.9821 CSS.CMS.Install.ConfigurationWizard.ViewModels.DatabaseViewModel [Error] -
An error occurred while preparing the database
at CSS.CMS.Install.ConfigurationWizard.ViewModels.DatabaseViewModel.b(Object A_0, RunWork-
erCompletedEventArgs A_1)

A connection was successfully established with the server, but then an error occurred during the
login process. (provider: SSL Provider, error: 0 - The certificate chain was issued by an authority
that is not trusted
```

Keyfactor Command version 10 requires an encrypted connection to the SQL server. If the SQL server is not configured correctly to receive a secure connection (is not configured with a valid certificate that is trusted by the Keyfactor Command server), you may receive this message.

For information about configuring TLS for SQL server, see:

> https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/enable-encrypted-connections-to-the-database-engine?view=sql-server-ver15

The request subject name is invalid or too long

```
The certificate request failed with the reason 'The request subject name is invalid or
too long. (Exception from HRESULT: 0x80094001).'
```

You may encounter this error on an enrollment when the CA rejects the request. If the request is clearly not excessively long, review the request for invalid characters. Be sure to also check the default subject (see the Subject Format application setting on the Application Settings: Enrollment Tab on page 609 and the subject defaults in certificate templates for both Configuring System-Wide Settings on page 382 and Enrollment Defaults Tab on page 398). Quotation marks should not be used in the fields of the default subject except in the case where these are part of the desired subject value, as they are processed as literal values. This is a change from earlier versions of Keyfactor Command where quotation marks were used around fields containing embedded commas.

This error can also appear if a Microsoft CA receives an enrollment request with no subject at all.

An error occurred while reading the key ring

You may encounter this error in the Keyfactor Command logs when trying to connect to the Management Portal. This can indicate that one or more of the service accounts configured to run the IIS application pools for Keyfactor Command do not have private key permissions to the certificate used to provide application-level encryption in Keyfactor Command (see Application-Level Encryption on page 2752 and Database Tab on page 2799). To resolve this, grant private key read permissions on that certificate to each application pool user.

```
Category: Microsoft.AspNetCore.DataProtection.KeyManagement.KeyRingProvider

EventId: 48

An error occurred while reading the key ring.

Exception:

System.Exception: Unable to decrypt enveloped PKCS7 data
```

Request failed with status code 405

You may encounter this error either in the Keyfactor CommandManagement Portal or when submitting a Keyfactor API request. This error is typically not accompanied by any error in the Keyfactor Command logs. This error can occur if the IIS *WebDAV Publishing* feature is installed on the Keyfactor Command server. Keyfactor Command is not compatible with this IIS feature. Uninstall the *WebDAV Publishing* feature, reboot if required, and try your command again.

Denied by Policy Module: Class is not licensed for use 0x80040112

This error may appear during certificate enrollment against a certificate authority running the Keyfactor CA Policy Module if the license for the policy module has expired. See License Expiration Monitoring and Rotation on page 817 for license update information.

Incorrect property 'value' for basic certificate extension

You may encounter an error similar to the following in the EJBCA log (not one of the Keyfactor Command logs) on a certificate enrollment if you make an enrollment attempt through Keyfactor Command using an EJBCA certificate profile configured with one or more custom certificate extensions that do not have a matching enrollment field configuration in Keyfactor Command. This may occur when using templates intended for use with the Keyfactor Windows Enrollment Gateway with incomplete Keyfactor Command configuration.

```
2023-07-05 19:43:45,248 ERROR
[org.cesecore.certificates.certificate.CertificateCreateSessionBean] (default task-19)
Error creating certificate:
org.cesecore.certificates.certificate.certextensions.CertificateExtensionException:
Incorrect property 'value' for basic certificate extension with id : 0 and OID :
1.3.6.1.4.1.311.21.7
```

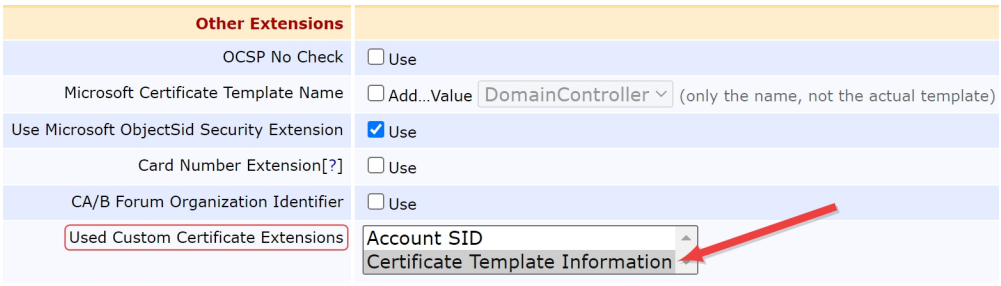


Figure 423: EJBCA Certificate Profile Custom Certificate Extensions

To operate correctly with Keyfactor Command, the profiles in EJBCA will need to be configured to not expect extension data. If you're using the the Keyfactor Windows Enrollment Gateway and wish to do direct enrollments in Keyfactor Command for your EJBCA CA as well as through the Keyfactor Windows Enrollment Gateway, you will need a separate set of certificate profiles in EJBCA that are not configured to expect extension data with the request to use for direct enrollment from Keyfactor Command.

Web Server Errors When Authenticating with an Identity Provider Other Than Active Directory

If you encounter a web server error (e.g. a 502 error) while authenticating with an identity provider other than Active Directory or across the entire Keyfactor Command product once you have authenticated with an identity provider other than Active Directory, it can be helpful to enable extended logging for the ClaimsProxy log. For more information, see Table 85: NLog.config Files for Keyfactor Command.

Certificate Validation Errors

On the Validation tab of the certificate details you will sometimes see a fail result for some of the validation tests. The following are some possible reasons why this might occur.

- If you see both *Full Chain* and *CRL Online* in a fail state, this generally indicates that you have not imported the root and/or intermediate certificate for the certificate into the appropriate store on the Keyfactor Command server (see Configure Certificate Chain Trusts for CAs on page 2763 in the *Keyfactor Command Server Installation Guide*).
- If you see just *CRL Online* in a fail state, this generally indicates that the Certificate Revocation List (CRL) for the CA could not be reached.

  Important: Because a "+" (plus sign) in a URL can represent either a space or a "+" Keyfactor Command has chosen to read "+" as a space. For CRL URLs that require a "+" (plus sign), rather than a space, replace plus signs in your CRL's URL with "`%2B`". Only replace the plus signs you don't wish to be treated as a space.

- If you see *Revocation Status* in a fail state but *CRL Online* is in a pass state, this can indicate that the CRL is accessible but expired, that the CRL is not fully configured, or that the Authority Information Access (AIA) for the CA has not been configured correctly or could not be reached.

For EJBCA CAs, CRLs and AIA need to be configured both at the CA level and at the certificate profile level. One way to do this is:

- AIA: Set the path to the AIA in the *CA issuer Default URI* field in the CA. You can find this on the *Fetch CA certificates* page of your EJBCA public web. Check both the *Authority Information Access* box and the *Use CA defined CA issuer* box in each certificate profile.
- CRL: Set the path to the CRL distribution point (CDP) in the *Default CRL Distribution Point* field in the CA. If appropriate for your environment, set also the *Default CRL Issuer* and/or *Default Freshest CRL Distribution Point* (delta CRLs). Check both the *CRL Distribution Points* box and the *Use CA defined CRL Distribution Point* box in each certificate profile.

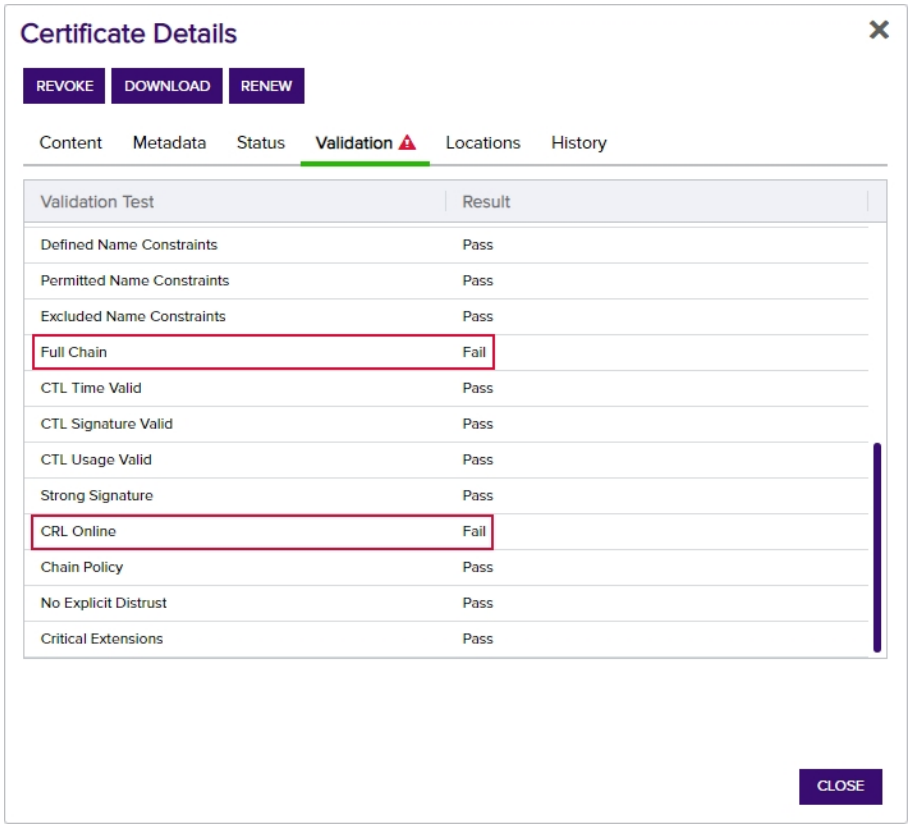


Figure 424: Certificate Validation Fails for Full Chain and CRL Online

Slow SSL Jobs

If SSL jobs are taking longer to complete than expected and you check the log on the orchestrator and find messages similar to the following:

```
2021-08-24 17:22:48.4463 Keyfactor.WindowsAgent.Jobs.SSL.SslDiscovery [Error] - Error
while sending SSL Batch for audit id 158558. Check the CMS Server log for more details.
Response status code does not indicate success: 413 (Request Entity Too Large).
at System.Net.Http.HttpResponseMessage.EnsureSuccessStatusCode()
at Keyfactor.WindowsAgent.Jobs.GenericJobExecutor`7.f.h()
```

```
2021-08-24 17:22:48.4463 Keyfactor.WindowsAgent.Jobs.SSL.SslDiscovery [Info] - Splitting
endpoint result batch of 29 into smaller pieces and retrying
```

You may want to make modifications to the IIS maximum request size settings on your Keyfactor Command server to allow it to accept larger incoming pieces of content to streamline SSL scanning. You can do this using the configuration editor built into the IIS management console. Make the setting changes at the Default Web Site level (or other web site, if you installed your Keyfactor Command in an alternate web site). There are three settings to change:

- system.webServer/security/requestFiltering/requestLimits/maxAllowedContentLength
- system.webServer/serverRuntime/uploadReadAheadSize
- system.web/httpRuntime/maxRequestLength

Set each system.webServer value to at least 1,000,000 bytes for best SSL scanning performance. The default value of 4096 KB for the maxRequestLength will probably be sufficient for SSL scanning in most environments, but if it has been reduced in your environment, you may need to increase it. (The system.webServer values are set in bytes while the system.web values are set in kilobytes.) If you are scanning networks with especially large numbers of returned certificates, you may need to increase all these values. Monitor the orchestrator logs after modifying the values to confirm that you have achieved the desired effect.

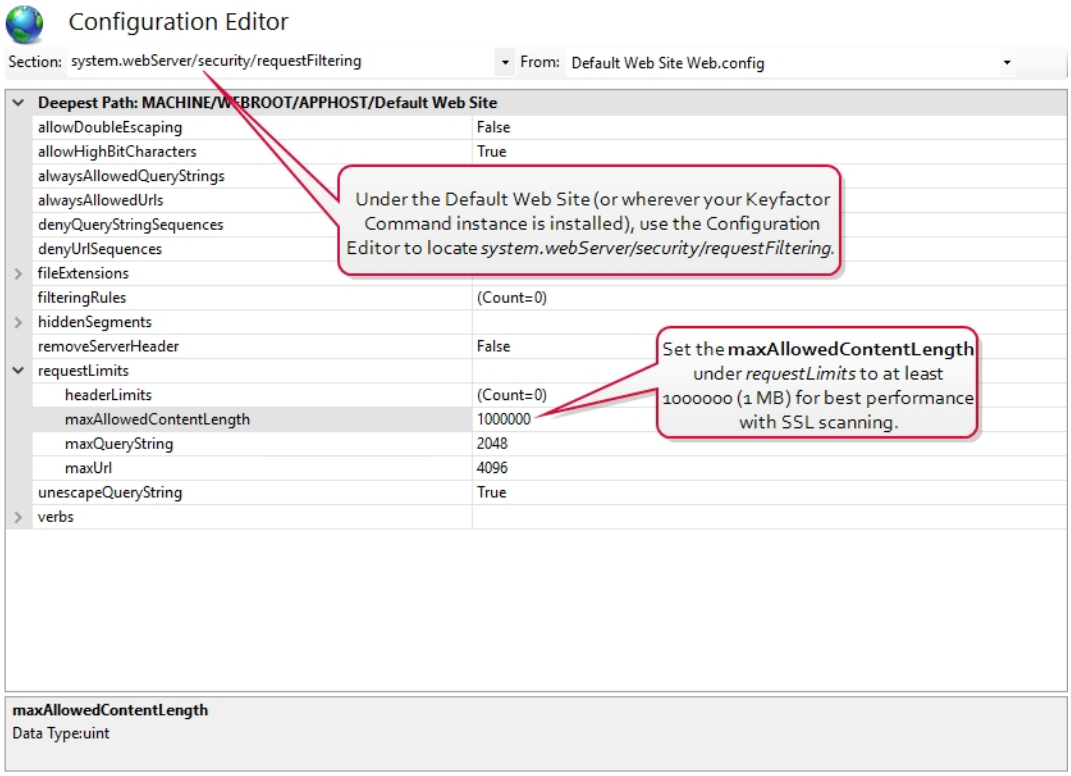


Figure 425: Modify IIS Settings for SSL Scanning: maxAllowedContentLength

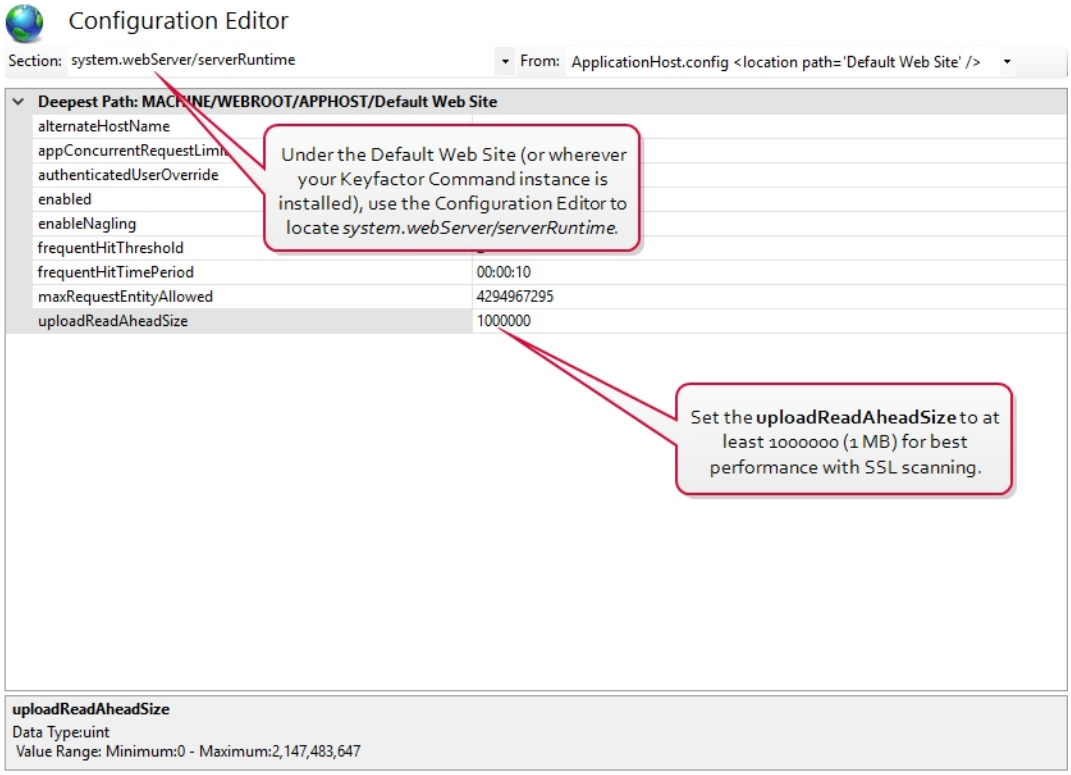


Figure 426: Modify IIS Settings for SSL Scanning:uploadReadAheadSize

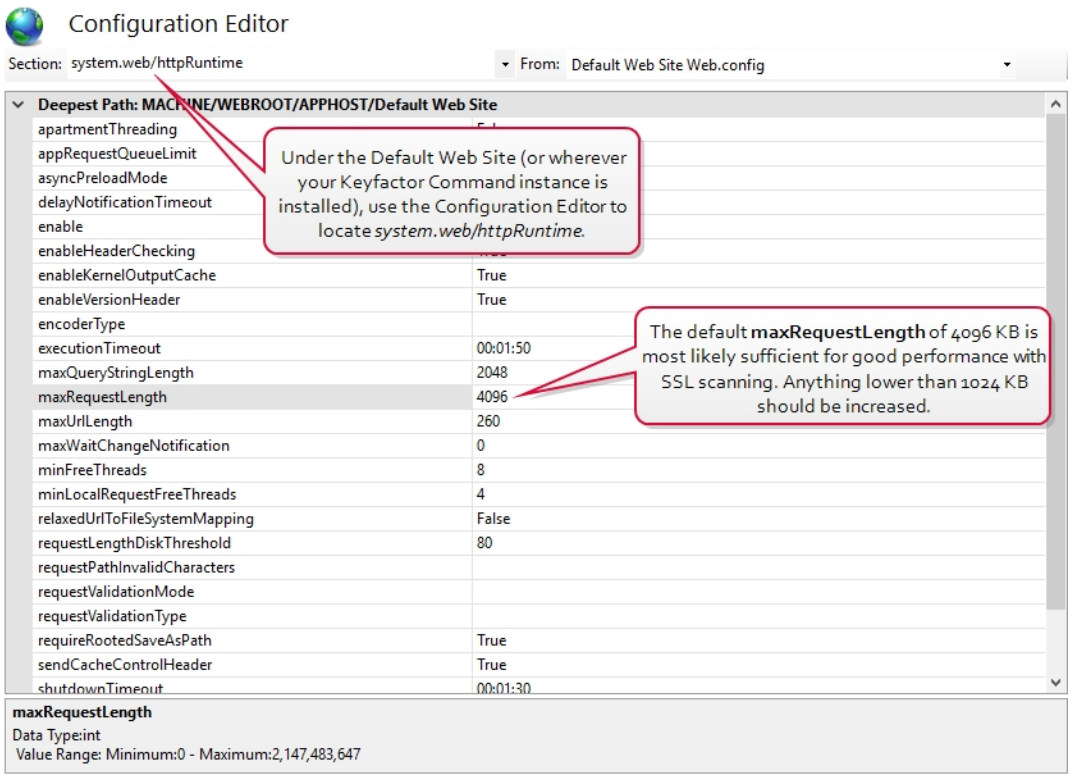


Figure 427: Modify IIS Settings for SSL Scanning: maxRequestLength

2.2.8.1 Disable Loopback Checking

For some features of the Management Portal to function correctly when using Kerberos authentication (e.g. delegation of CA functions, alerting using the event logging event handler and a DNS alias or alternate target machine), it may be necessary to disable loopback checking on the Keyfactor Command server.

To disable loopback checking for selected FQDNs:

1. On the Keyfactor Command server, open the registry editor and browse to:

   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters

2. Right-click the Parameters registry key and choose **New > DWORD (32-bit) Value**. Name the new DWORD value **DisableStrictNameChecking**. Set the **DisableStrictNameChecking** value to 1.

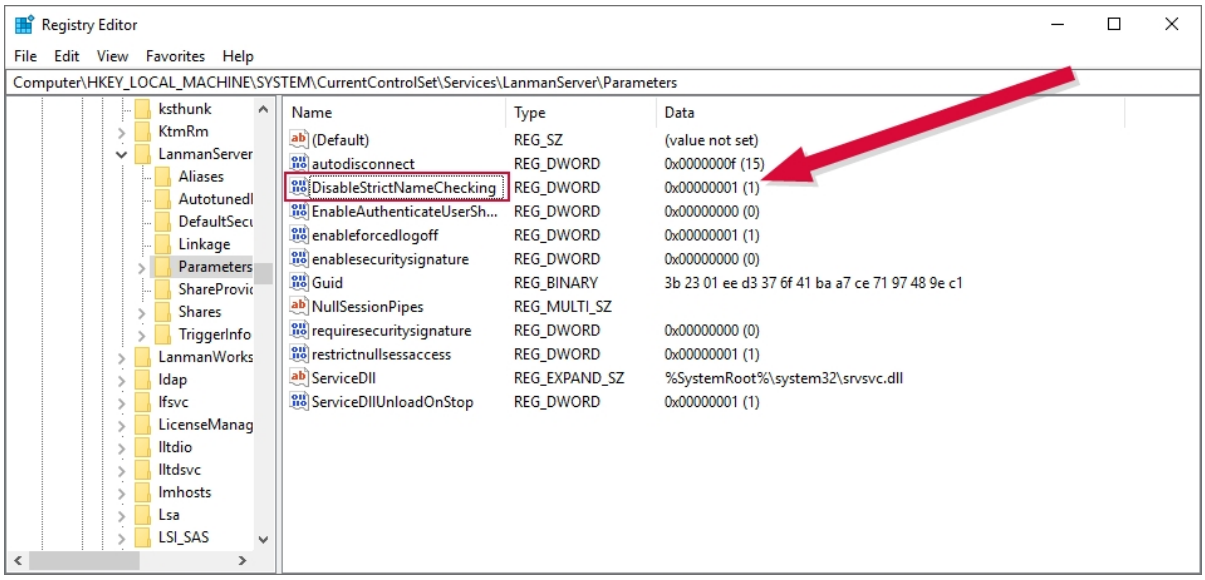


Figure 428: Disable Loopback Checking: DisableStrictNameChecking

3. In the registry editor browse to:

   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\MSV1_0

4. Right-click the MSV1_0 registry key and choose **New > Multi-String Value**. Name the new value **BackConnectionHostNames**. Edit the **BackConnectionHostNames** value and enter each fully qualified domain name—actual name or DNS alias—for a server that needs this feature on a separate line. For example, for full DNS alias support with CA delegation functions, you need to enter the DNS alias of the Keyfactor Command server. For event logging to a machine other than the Keyfactor Command server, you need to enter the name of that server.

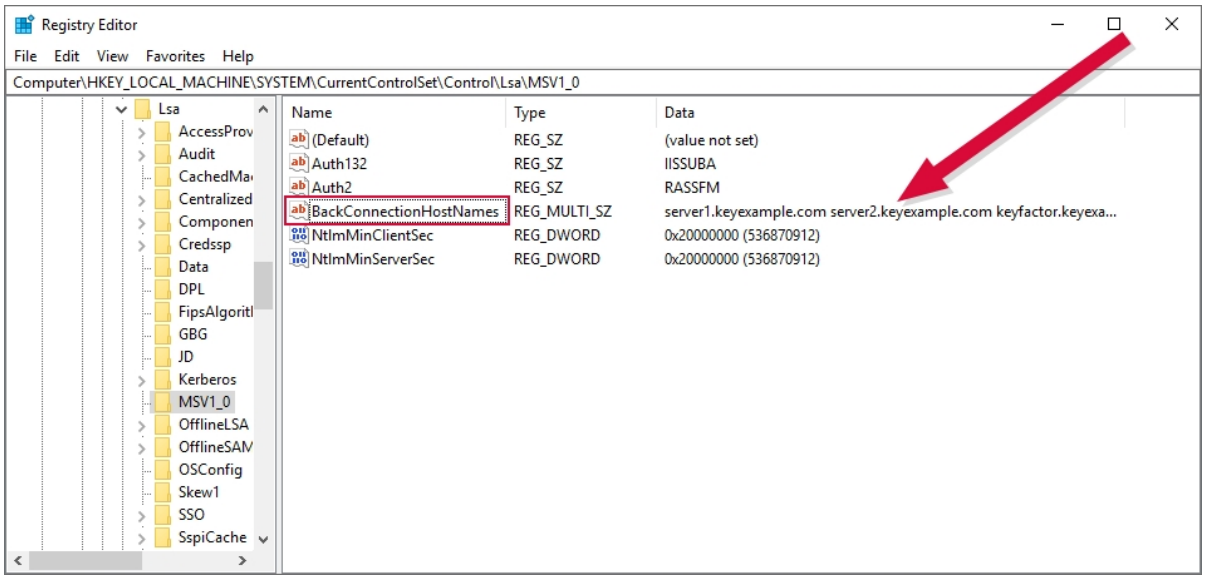


Figure 429: Disable Loopback Checking: BackConnectionHostNames

5. After completing the registry configuration you must reboot the Keyfactor Command server before the changes will take effect.

2.3 Appendices

- Appendix - References below
- Appendix - Third-Party Notices for Keyfactor Command Software below

2.3.1 Appendix - References

CIDR, Classless Inter-Domain Routing

http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

2.3.2 Appendix - Third-Party Notices for Keyfactor Command Software

This Software from Keyfactor incorporates or interacts with third-party material from the files listed below. While Keyfactor is not the original author of the third-party material, Keyfactor licenses this material under the terms set forth in the license agreements below.

Keyfactor Command distributions may include the following Third-Party Materials. Since many of these materials use the same copyright text, a copy of the applicable text from each license is provided below.

Table 89: Third-Party Notices for Keyfactor Command Software Distributions

Description	Version	Copyright Holder	License
ADOObjectPicker	1.0.0	Tulpep	Microsoft Public
ajaxFileInput	1.0.0	OpenJs	MIT
Apache Codec	1.6	Apache.org	Apache 2.0
Apache Commons	4.3.3	Apache.org	Apache 2.0
Apache http client	4.3.6	Apache.org	Apache 2.0
at-caret	1.3.1	Gideon Sireling	BSD
BouncyCastle	1.8.1	BouncyCastle	MIT
Chosen	1.0.0	Patrick Filler	MIT
Common Logging	3.2.0	(Multiple)	Apache 2.0
contextmenu	1.1	Matt Kruse	MIT
DateTimeEntry	2.0.0	Keith Wood	MIT
Filedownload	1.4.2	John Culviner	MIT
Flexigrid	1.1	Paolo Marinas	MIT
History.js	1.8b2	Community	BSD
Iframe	1.8.2	Sebastion Tschan	MIT
Joda Time	2.8.1	Apache.org	Apache 2.0
jqPlot	1.0.8	Chris Leonello	MIT
jQuery	2.1.0	jQuery Foundation	MIT
jQuery UI	1.10.3	jQuery Foundation	MIT
jQuery Validate	1.9	Jorn Zaeffer	MIT
jsTree	3.1.0	Ivan Bozhanov	MIT
Layout	1.3.0	Kevin Dalman	MIT
Log4j2	2.1	Apache.org	Apache 2.0

Description	Version	Copyright Holder	License
NewtonSoft	6.0.8	James Newton-King	MIT
NLog	4	(Multiple)	MIT
Quartz	2.3.3	Marko Lahama	Apache 2.0
Unity	4.0.1	Microsoft	Microsoft Public
WiX	3.1	.NET Foundation	Microsoft Reciprocal
WPF Extensions	2.2.0	Microsoft	Microsoft Public

A copy of the applicable text from each license is provided below.

2.3.2.1 Apache 2.0 License Text:

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

“License” shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

“Licensor” shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

“Legal Entity” shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, “control” means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

“You” (or “Your”) shall mean an individual or Legal Entity exercising permissions granted by this License.

“Source” form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

“Object” form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

“Work” shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

“Derivative Works” shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

“Contribution” shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, “submitted” means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as “Not a Contribution.”

“Contributor” shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and

You must cause any modified files to carry prominent notices stating that You changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work

stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

2.3.2.2 BSD License Text:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2.3.2.3 MIT License Text:

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2.3.2.4 Microsoft Public License (MS-PL) Text:

This license governs use of the accompanying software. If you use the software, you accept this license. If you do not accept the license, do not use the software.

1. Definitions

The terms “reproduce,” “reproduction,” “derivative works,” and “distribution” have the same meaning here as under U.S. copyright law.

A “contribution” is the original software, or any additions or changes to the software.

A “contributor” is any person that distributes its contribution under this license.

“Licensed patents” are a contributor’s patent claims that read directly on its contribution.

2. Grant of Rights

(A) Copyright Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free copyright license to reproduce its contribution, prepare derivative works of its contribution, and distribute its contribution or any derivative works that you create.

(B) Patent Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free license under its licensed patents to make, have made, use, sell, offer for sale, import, and/or otherwise dispose of its contribution in the software or derivative works of the contribution in the software.

3. Conditions and Limitations

(A) No Trademark License- This license does not grant you rights to use any contributors’ name, logo, or trademarks.

(B) If you bring a patent claim against any contributor over patents that you claim are infringed by the software, your patent license from such contributor to the software ends automatically.

(C) If you distribute any portion of the software, you must retain all copyright, patent, trademark, and attribution notices that are present in the software.

(D) If you distribute any portion of the software in source code form, you may do so only under this license by including a complete copy of this license with your distribution. If you distribute any portion of the software in compiled or object code form, you may only do so under a license that complies with this license.

(E) The software is licensed “as-is.” You bear the risk of using it. The contributors give no express warranties, guarantees or conditions. You may have additional consumer rights under your local laws which this license cannot change. To the extent permitted under your local laws, the contributors exclude the implied warranties of merchantability, fitness for a particular purpose and non-infringement.

2.3.2.5 Microsoft Reciprocal License (MS-RL) Text:

This license governs use of the accompanying software. If you use the software, you accept this license. If you do not accept the license, do not use the software.

1. Definitions

The terms “reproduce,” “reproduction,” “derivative works,” and “distribution” have the same meaning here as under U.S. copyright law.

A “contribution” is the original software, or any additions or changes to the software.

A “contributor” is any person that distributes its contribution under this license.

“Licensed patents” are a contributor’s patent claims that read directly on its contribution.

2. Grant of Rights

(A) Copyright Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free copyright license to reproduce its contribution, prepare derivative works of its contribution, and distribute its contribution or any derivative works that you create.

(B) Patent Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free license under its licensed patents to make, have made, use, sell, offer for sale, import, and/or otherwise dispose of its contribution in the software or derivative works of the contribution in the software.

3. Conditions and Limitations

(A) Reciprocal Grants- For any file you distribute that contains code from the software (in source code or binary format), you must provide recipients the source code to that file along with a copy of this license, which license will govern that file. You may license other files that are entirely your own work and do not contain code from the software under any terms you choose.

(B) No Trademark License- This license does not grant you rights to use any contributors’ name, logo, or trademarks.

(C) If you bring a patent claim against any contributor over patents that you claim are infringed by the software, your patent license from such contributor to the software ends automatically.

(D) If you distribute any portion of the software, you must retain all copyright, patent, trademark, and attribution notices that are present in the software.

(E) If you distribute any portion of the software in source code form, you may do so only under this license by including a complete copy of this license with your distribution. If you distribute any portion of the software in compiled or object code form, you may only do so under a license that complies with this license.

(F) The software is licensed “as-is.” You bear the risk of using it. The contributors give no express warranties, guarantees or conditions. You may have additional consumer rights under your local laws which this license cannot change. To the extent permitted under your local laws, the contributors exclude the implied warranties of merchantability, fitness for a particular purpose and non-infringement.

3.0 Keyfactor API Reference

The Keyfactor Command solution exposes an API to allow third-party software to integrate with the advanced certificate enrollment and management features of Keyfactor Command in a secure manner and to provide a mechanism for automating routine or bulk tasks that would be cumbersome to perform through the browser-based user interface. The API complements the web components of Keyfactor Command and offer a number of HTTP method calls that provide similar functionality to that available within the Management Portal's user interface, but which can be accessed programmatically by any system capable of making web requests. The API has the following goals and constraints:

- Provide a simple interface to make integration easy for third parties.
- Develop interoperability between different technology frameworks and operating systems.
- Support common certificate enrollment and management tasks.
- Deliver a securable interface.
- Preserve backward-compatibility so that existing clients continue to work, where possible.

Important: The Classic API, also known as the CMS API, was deprecated in Keyfactor Command version 11. All uses of the Classic API should be migrated to the Keyfactor API prior to upgrading to Keyfactor Command version 11.

3.1 Overview

In the current release, Keyfactor exposes one API for external use. The Keyfactor API was introduced in Keyfactor Command version 6.1 and as of Keyfactor Command 11.0 is the only supported API. The Keyfactor API allows for integration with other systems to automate certificate lifecycle management tasks. It will continue to be developed going forward to expose more core functionality that is built into the main product to allow for more in-depth integrations.

Documentation for the Keyfactor API is available as two companion pieces—this document (the *Keyfactor API Reference Guide*), which provides an overview of the API's endpoints, parameters to be provided in them, and data expected back from them, and the interactive code examples installed with your Keyfactor Command instance in the *Keyfactor API Reference and Utility*.

Important: The Classic API, also known as the CMS API, was deprecated in Keyfactor Command version 11. All uses of the Classic API should be migrated to the Keyfactor API prior to upgrading to Keyfactor Command version 11.

3.2 Authenticating to the Keyfactor API

When you make a connection to Keyfactor Command using the Keyfactor API, you need to provide authentication. If you're using Active Directory as an identity provider, you have the choice to authenticate to Keyfactor Command using Basic authentication or Windows integrated authentication. Any users who have already authenticated to Keyfactor Command before opening the Keyfactor API Reference and Utility in the same browser session will be seamlessly authenticated to Keyfactor Command automatically, and will not need to re-authenticate. The need to intentionally provide authentication for the Keyfactor API comes into play in situations such as:

- You are developing or running an application or script that leverages the Keyfactor API.
- You are running a workflow step of type *Invoke REST Request* with Active Directory Basic or Windows authentication.
- You are running a workflow step of type *Invoke REST Request with OAuth* with an identity provider other than Active Directory and token authentication.
- You are using the Keyfactor API Reference and Utility and using an identity provider other than Active Directory (see Acquire a Token to Authenticate to the Keyfactor API on the next page).
- You are using the Keyfactor API Reference and Utility, using Active Directory as an identity provider, and have not authenticated to Keyfactor Command within the same browser session.

In many of these cases, you will probably want to make the API requests not as an individual user, but as a service account. The service account you use depends on the identity provider you're using:

- If you're using Active Directory as an identity provider, a standard Active Directory service account in the primary Keyfactor Command server forest can be used.
- If you're using an identity provider other than Active Directory, a client (not user) in your identity provider is used. The client should be configured with a secret and have *Client authentication* and *Service account roles* enabled (see Service Accounts on page 2730). The user who will make use of the API will need the client ID and secret as well as the bearer token URL. This is the URL of the token endpoint for your identity provider instance. For example:

  ```
  https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token
  ```

  For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

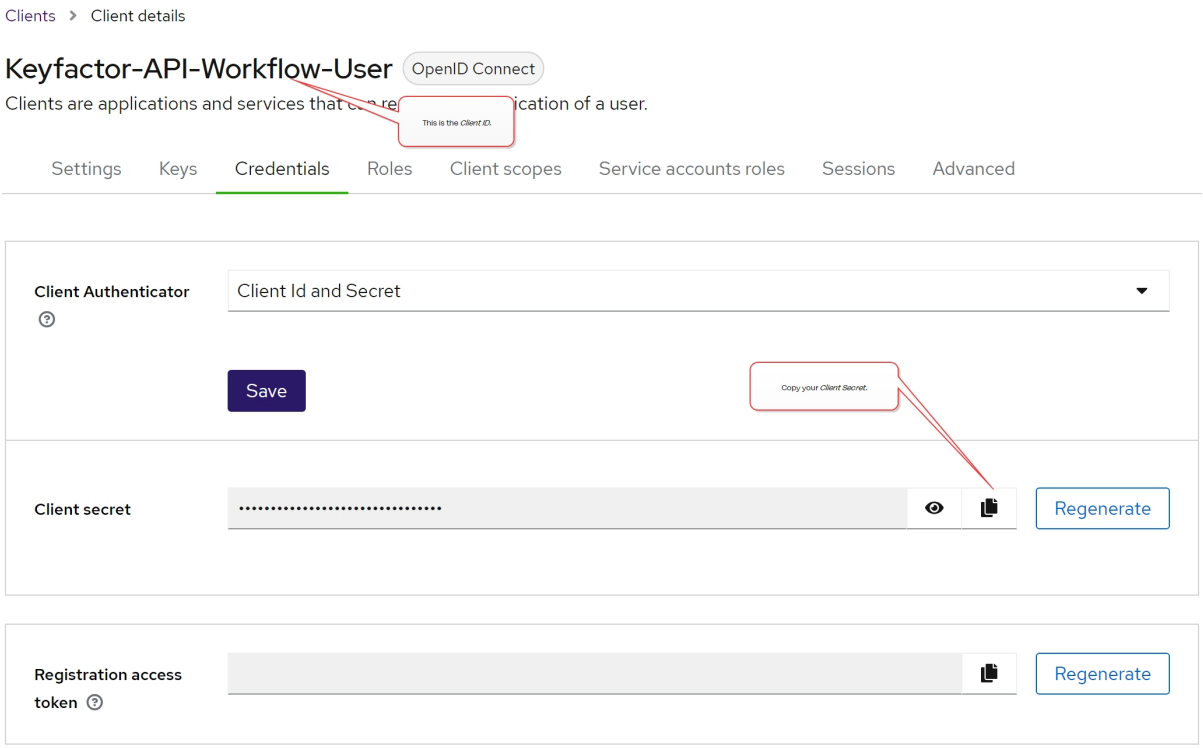


Figure 430: Client Secret for Keyfactor API in Keyfactor Identity Provider

This service account needs to be granted appropriate permissions in Keyfactor Command to complete the API requests that will be run as this service account.

Acquire a Token to Authenticate to the Keyfactor API

If you're using the Keyfactor API Reference and Utility (Swagger) and using an identity provider other than Active Directory, you will need to acquire a token from your identity provider in order to authenticate to the Keyfactor API Reference and Utility. There are a number of approaches to doing this. Here we provide a couple of examples.

First, be sure that you have created a client in your identity provider (see Service Accounts on page 2730) and that you know this information about the client:

- Client ID
- Client Secret
- Bearer Token URL

To acquire a token to authenticate to the Keyfactor API either via the Keyfactor API Reference and Utility or directly, from a Linux server execute a curl command similar to the following, referencing appropriate values for your client ID, client secret, and bearer token URL:

```
curl --request POST --url https://appsrvr18.keyexample.com:1443/realms/Keyfactor/protocol/openid-
connect/token --header 'Content-Type: application/x-www-form-urlencoded' --data client_
secret=o2mwI5LjZEAmcbC9dnhHq589BOf2OqD4 --data client_id=Keyfactor-API-Workflow-User --data grant_
type=client_credentials
```

To acquire a token to authenticate to the Keyfactor API either via the Keyfactor API Reference and Utility or directly, in PowerShell execute a script similar to the following, referencing appropriate values for your client ID, client secret, and bearer token URL:

```
$Body = @{
   grant_type = "client_credentials"
   client_id = "Keyfactor-API-Workflow-User"
   client_secret = "o2mwI5LjZEAmcbC9dnhHq589BOf2OqD4"
}

$Headers = @{
   'Content-Type' = 'application/x-www-form-urlencoded'
}

$TokenResults = Invoke-RestMethod -Method Post -Uri https://appsrvr18.keyexample.com:1443/realms/Key-
factor/protocol/openid-connect/token -Headers $Headers -Body $Body

# Output the token string to a file to avoid CR/LFs
$MyToken = $TokenResults.access_token
Set-Content -Value $MyToken -Path C:\Stuff\MyTokenOutFile.txt
```

In both cases, the results will include an *access_token* value in addition to other data, as shown in Figure 431: Access Token for the Keyfactor API Reference and Utility. In the PowerShell case, this is output to a file to avoid introducing spaces or line wraps into the token value. You can open the output file, display the token without line wrapping, and copy the token value for pasting into the Keyfactor API Reference and Utility. Any line wraps that display on the screen in either the PowerShell window or text editor window will be interpreted by copy/paste as CR/LF, which will cause API commands to fail when the resulting token is submitted. Be sure to use a text editor to open the output file that can display the entire length of the token string as a single line. The built-in Windows Notepad application will display a maximum of 1024 characters on a line before wrapping even if word wrap is disabled. A tool such as the third-party Notepad++ is much less limited.

```
access_token       : eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJmYU04NjQ5UHAzREl1NWNBWkk4a3NYSVRYek0xV2sxb3p5WmRFZ3VPUzBJIn0.eyJleHAiOjE2OTUxNTQyMTMsImlhdCI6MTY5NTE1MzkxMywianRpIjoiMjVkN2RmMWU
                     tYWEyNy00MThjLTkwOTYtNjIzMTUyYjcwNDViIiwiaXNzIjoiaHR0cHM6Ly9hcHBzcnZyMTg2LmtleWV4YW1wbGUuY29tOjM0NDMvcmVhbG1zL0tleWZhY3RvciIsImF1ZCI6ImFjY291bnQiLCJzdWIiOiI5OTI4NmI5My1hNjcyLTQ3Yj
                     ktYTgwZS0xNDQ3YjMyZWEzOTciLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiIxMjMtNDU2IiwiYWNyIjoiMSIsImFsbG93ZWQtb3JpZ2lucyI6WyIvKiIsImh0dHBzOi8va2V5ZmFjdG9yMjYuYnVmZnkuY29tOjQ0NCJdLCJyZXNvdXJjZV9hY
                     2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hbmFnZS1hY2NvdW50LWxpbmtzIiwidmlldy1wcm9maWxlIl19fSwic2NvcGUiOiJlbWFpbCBwcm9maWxlIiwiY2xpZW50SG9zdCI6IjEwLjQuMy4yNiIs
                     ImVtYWlsX3ZlcmlmaWVkIjpmYWxzZSwicm9sZSI6WyJkZWZhdWx0LXJvbGVzLWtleWZhY3RvciIsIm9mZmxpbmVfYWNjZXNzIiwidW1hX2F1dGhvcml6YXRpb24iXSwicHJlZmVycmVkX3VzZXJuYW1lIjoic2VydmljZS1hY2NvdW50LTE
                     yMy00NTYiLCJjbGllbnRBZGRyZXNzIjoiMTAuNC4zLjI2IiwiY2xpZW50X2lkIjoiMTIzLTQ1NiJ9.WQg9GFYCW-JmSjqiSaOxjwHxiV2JzLEcOZRwv2KFg5mAH1mLpQtzgfiJ4uVIFJgs05pQck1fLUl6OIF1Ofp7gdCkrgqst5kmgL3U4
                     FjOw0KmIoPWfTu_APhMltlbXnnO8NCic2TfaF_IrVjlc95EK4b6IaEbwcbCIg_896f5bOxrXedouGP12dbRHOqB2jffD1glDveTB2XL4WnbTVuSbgc2NsISoNzGZB-HGXIWllo4l-PXK42nY5YUr7kOlf2W39HSojkyJRuwrpBjeVUmeDVQ
                     _njCQ1rufxrDK1ZkAnbw3rYiJKGzsVJzAlNwTFiM6-9pHPz68Nc1rPwviPyAmQ
expires_in         : 300
refresh_expires_in : 0
token_type         : Bearer
not-before-policy  : 0
scope              : email profile
```

Figure 431: Access Token for the Keyfactor API Reference and Utility

To use the token in the Keyfactor API Reference and Utility:

1. Copy the access token value only with no spaces or CR/LFs. For example:

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJmYU04NjQ5UHAzREl1NWNBWkk4a3NYSV
[portion removed for display ] KmIoPWfTu_APhMltlbXnnO8NCic2TfaF_
IrVjlc95EK4b6IaEbWcbCIg_
896f5bOxrXedouGP12dbRHOqB2jffD1glDveTB2XL4WnbTVuSbgc2NsISoNzGZB-HGXIWllo4l-
PXK42nY5YUr7k0lf2W39HSojkyJRuwrpBjeVUmeDVQ_njCQ1rufxrDK1ZkAnbw3rYiJKGzsVJzAlNwTFiM6-
9pHPz68Nc1rPwviPyAmQ
```

2. In the Keyfactor API Reference and Utility click either the **Authorize** button at the top or one of the padlock authorization icons on each endpoint to open the authorization dialog.

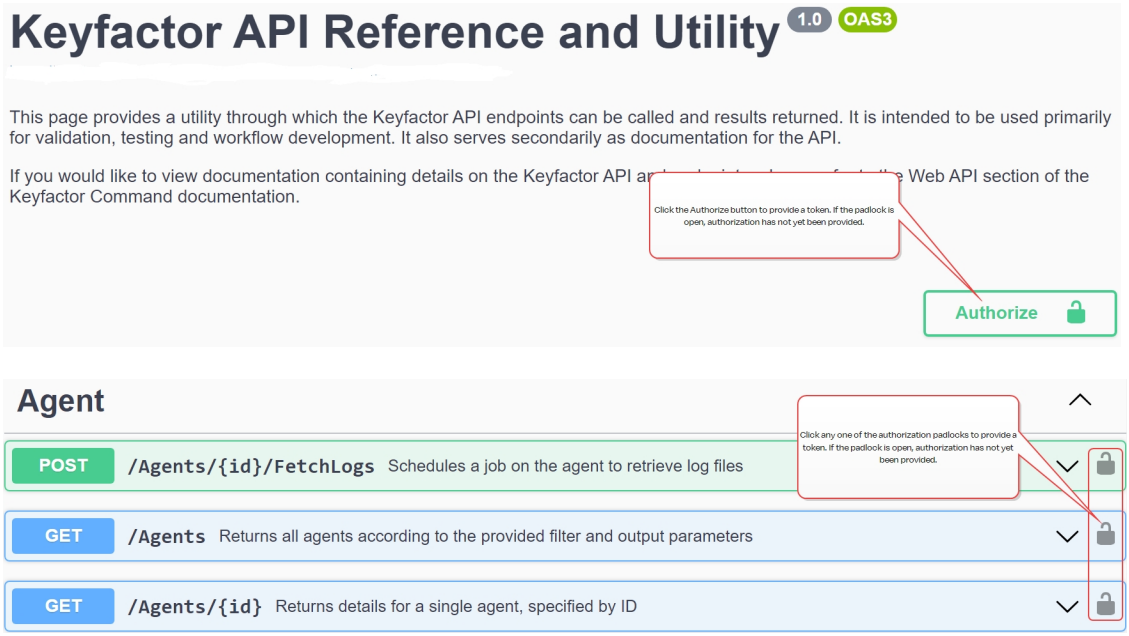


Figure 432: Keyfactor API Reference and Utility Authorize Options

3. In the Available authorizations dialog, paste in your access token value and click **Authorize**.

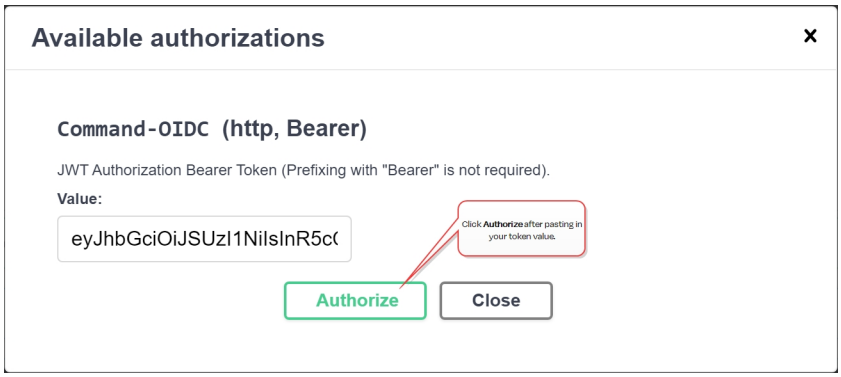


Figure 433: Enter Access Token in the Keyfactor API Reference and Utility

4. If authorization is successful, the Authorize button will change to *Logout*, and the padlocks will change to locked.

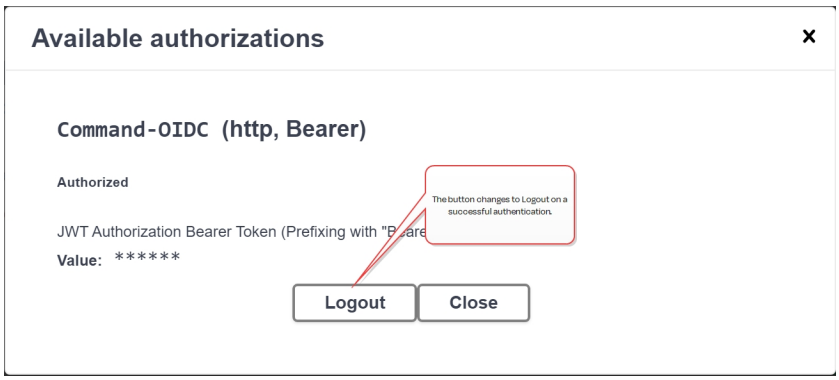


Figure 434: Successful Authorization in the Keyfactor API Reference and Utility

Tip: When using a token in the Keyfactor API Reference and Utility, you use the token value only. When you use a token to authenticate to the Keyfactor API other than through the Keyfactor API Reference and Utility, you need to precede the token value with *Bearer*. For example:

```
# Build the headers for the API request
$headers = @{
   "Authorization"="Bearer " + $TokenValue.access_token
   "Accept"="application/json"
   "x-keyfactor-requested-with"="APIClient"
}
```

For an example script using a token to authenticate to Keyfactor Command, see Use Custom PowerShell with Embedded REST Request, Send Email, and Require Approval on page 266.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

3.3 Transaction Security

The Keyfactor API relies on SSL/TLS to protect the HTTP communications between the client and Keyfactor Command server. In a typical deployment, the API will be configured for Basic authentication, where client credentials are provided in an HTTP header, formatted as *DOMAIN\user-:Password* and base-64-encoded. Basic Authentication itself is not a secure way to pass a set of user credentials. However, it is very interoperable and works well across all of the various technologies that use the API. SSL is used to protect the confidentiality of user credentials; therefore, SSL should be used with the Keyfactor API.

Keyfactor recommends that any device using the API already be configured to trust the SSL certificate presented by Keyfactor Command, allowing the SSL connection to be established without error. The process for this will depend on the platform and operating environment of the connecting client, but the appropriate documentation or support for your platform should outline the necessary steps for this.

Access to the API methods can be limited per client to a maximum request frequency. The amount of time required between calls can be configured in the Keyfactor Command Management Portal Application Settings for the API (see Application Settings: API Tab on page 619). Increasing this interval can mitigate certain threats such as denial of service or dictionary attacks against passwords and other sensitive data. However, setting this too high can negatively impact performance of client applications that need to make a large number of requests.

3.4 Endpoint Common Features

Some aspects of the Keyfactor API request and response formats are consistent across all endpoints. This includes a small set of HTTP headers, HTTP statuses returned by the server for successful requests, and various error conditions. Common request headers are given in Table 90: Common Request Headers, common response headers (for successful requests and certain unsuccessful requests) are given in Table 91: Common Response Headers, and HTTP statuses are given in Table 92: HTTP Statuses.

By default, all Keyfactor API methods start with a base path, which corresponds to an application under IIS; this path is configurable at install time. The default base path is *KeyfactorApi*. The API component name and method name then comprise the parts of the URL, each separated by a forward slash. For example, */KeyfactorApi/Certificates/Import* would be the URL format for the Import method of the Certificates component.

Table 90: Common Request Headers

Header Name	**API Version**	Header Value	Description
Content-Type	Both	application/json OR application/xml	POST methods use application/json. When application/xml is needed, it is specifically indicated on the endpoint page.
Accept	Both	application/json; char-set=utf-8	Most methods returning complex values will use this content type.
Authorization	Both	Basic <base-64 DOMAIN\user:pass>	In most cases, Web API clients will use Basic authentication over SSL/TLS.
Host	Both	<Keyfactor Command server hostname>	Address of Keyfactor Command server. Automatically generated in most clients.
Content-Length	Both	Request length in bytes	Optional, but automatically gener-ated by most clients.
X-Keyfactor-Requested-With	Both	XMLHttpRequest	This is mandatory to send in a request to the Keyfactor API on POSTs, PUTs, and DELETEs, and the value is case sensitive. This is for security.
X-Keyfactor-API-Version	Keyfactor API	1 or 2	Desired version of the endpoint. If not provided, this defaults to version 1.

Table 91: Common Response Headers

Header Name	Header Value	Description
Cache-Control	no-cache	API requests are generally not cacheable. Note that this is not respected by all client systems.
Pragma	no-cache	API requests are generally not cacheable. Note that this is not respected by all client systems.
Content-Length	<varies>	Length of the HTTP response.
Content-Type	application/json	Most calls return application/json, but occasionally text/-

Header Name	Header Value	Description
		plain or text/xml.
Expires	-1	Usually ignored.
Server	<varies>	Software version reported by IIS platform hosting Keyfactor Command.
X-Keyfactor-Product-Version	<varies>	Keyfactor Command platform version.
X-Total-Count	<varies>	Total number of elements returned.
X-AspNet-Version	<varies>	Version of ASP.NET supporting Keyfactor Command installation.
X-Powered-By	ASP.NET	Header added by underlying ASP.NET implementation.
Date	<varies>	Timestamp of the HTTP response.

Table 92: HTTP Statuses

Number/Name	Description
200 OK	Request successful; results in response body
204 No Content	Request successful; no content in response body
400 Bad Request	Malformed or invalid data; additional information may be available in the response body and/or Keyfactor Command server logs
401 Unauthorized	Invalid credentials (user unauthenticated)
403 Forbidden	Can often indicate that the credentials map to a user without permissions for this action in Keyfactor Command (user unauthorized)
404 Page not Found	Invalid request path
500 Internal Server Error	Keyfactor Command encountered an unexpected error attempting to handle the request. See response body and Keyfactor Command server logs for details.
502 Bad Gateway	Keyfactor Command attempted to contact a CA or other upstream server to process the request, but was unable to. See Keyfactor Command server logs for details.

3.5 Versioning

The Keyfactor API is versioned as a set and released in conjunction with Keyfactor Command at the same version level (e.g. version 11.1). In addition, the Keyfactor API may have multiple versions of select endpoints.

The current strategy is to increment the version of an API when changes are made that might break backwards compatibility for existing clients. New endpoints are generally implemented in the most recent version of their API.

Generally, updates to an existing version of an endpoint are restricted to updates that should not break existing clients. Updates may be made that add HTTP response headers or response body parameters, or that correct existing bugs, or must be made to conform to newer or more granular security constraints. When an update cannot be made without breaking existing clients, a new endpoint is added in a later API version.

Figure 435: Select a Version in the Keyfactor API Reference and Utility

Most Keyfactor API endpoints have only one version, though a second version has been released for a select few endpoints. The Keyfactor API uses the *x-keyfactor-api-version* request header to differentiate between versions 1 and 2 of a given endpoint. If a version isn't specified, version 1 is assumed.

Important: The Classic API, also known as the CMS API, was deprecated in Keyfactor Command version 11. All uses of the Classic API should be migrated to the Keyfactor API prior to upgrading to Keyfactor Command version 11.

3.6 Keyfactor API Endpoints

The documentation for the Keyfactor API endpoints in the following sections includes descriptions for all the parameters available for each endpoint and short examples of specific parameters where that has been deemed to be helpful. For complete usage examples, see the Keyfactor API Reference and Utility.

Tip: Click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Logout** button to find the embedded web copies of the *Keyfactor Command*

Documentation Suite and the *Keyfactor API Reference and Utility*.

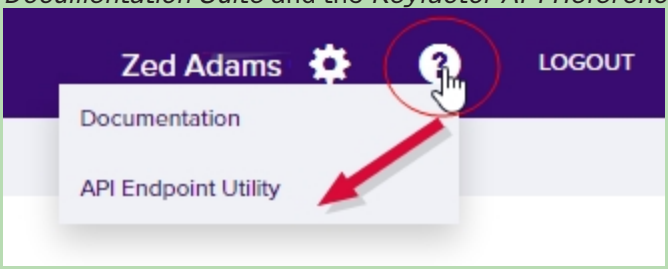


Figure 436: Documentation in the Help Dropdown

You can also browse to the *Keyfactor API Reference and Utility* directly using the following link (where *keyfactor.keyexample.com* is the fully qualified domain name of your Keyfactor Command server or the DNS alias you are using to reference your Keyfactor Command server, if applicable):

```
https://keyfactor.keyexample.com/KeyfactorAPI/ref/index#
```

```
https://keyfactor.keyexample.com/KeyfactorAPINET6/swagger
```

This link assumes that the Keyfactor API has been installed in the default IIS virtual directory (KeyfactorAPI). If you have installed in an alternate virtual directory, your path will be different.

3.6.1 Agents

The Agents component of the Keyfactor API includes methods necessary to list orchestrators and agents and schedule jobs to retrieve log files for orchestrators and agents that support that functionality.

Table 93: Agents Endpoints

Endpoint	Method	Description	Link
/{id}	GET	Returns details for a single orchestrator or agent.	GET Agents ID on the next page
/	GET	Returns a list of all orchestrators and agents according to the provided filters and input parameters.	GET Agents on page 858
/Reset	POST	Resets one or more orchestrators or agents to a new state and clears jobs.	POST Agents Reset on page 864

Endpoint	Method	Description	Link
/Approve	POST	Approves an orchestrator.	POST Agents Approve on page 864
/Disapprove	POST	Disapproves an orchestrator.	POST Agents Disapprove on page 865
/{id}/Reset	POST	Resets a single orchestrator or agent to a new state and clears jobs.	POST Agents ID Reset on page 866
/{id}/FetchLogs	POST	Schedules a job on the orchestrator or agent to retrieve log files.	POST Agents ID FetchLogs on page 866
/SetAuthCertificateReenrollment	POST	Configures an orchestrator or agent to either request or require a new client authentication certificate on its next session registration.	POST Agents Set Auth Certificate Reenrollment on page 867

3.6.1.1 GET Agents ID

The GET /Agents/{id} method is used to retrieve a single orchestrator or agent registered in Keyfactor Command. This method returns HTTP 200 OK on a success with a list of all orchestrator details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 94: GET Agents{id} Input Parameters

Name	In	Description
id	Path	**Required**. The GUID of the orchestrator to retrieve. Use the *GET /Agents* method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Table 95: GET Agent {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>AgentId</td><td>A string indicating the GUID of the orchestrator.</td></tr>
<tr><td>ClientMachine</td><td>A string indicating the client machine on which the orchestrator is installed.</td></tr>
<tr><td>Username</td><td>A string indicating the Active Directory user or service account the orchestrator is using to connect to Keyfactor Command.</td></tr>
<tr><td>AgentPlatform</td><td>An integer indicating the platform for the orchestrator. Possible values are:
<table>
<tr><th>Value</th><th>Parameter Value</th></tr>
<tr><td>0</td><td>Unknown</td></tr>
<tr><td>1</td><td>Keyfactor Windows Orchestrator</td></tr>
<tr><td>2</td><td>Keyfactor Java Agent</td></tr>
<tr><td>3</td><td>Keyfactor Mac Auto-Enrollment Agent</td></tr>
<tr><td>4</td><td>Keyfactor Android Agent</td></tr>
<tr><td>5</td><td>Keyfactor Native Agent</td></tr>
<tr><td>6</td><td>Keyfactor Bash Orchestrator</td></tr>
<tr><td>7</td><td>Keyfactor Universal Orchestrator</td></tr>
</table></td></tr>
<tr><td>Version</td><td>A string indicating the version of the orchestrator.</td></tr>
<tr><td>Status</td><td>An integer indicating the orchestrator status. Possible values are:
<table>
<tr><th>Value</th><th>Parameter Value</th></tr>
<tr><td>1</td><td>New</td></tr>
<tr><td>2</td><td>Approved</td></tr>
<tr><td>3</td><td>Disapproved</td></tr>
</table></td></tr>
<tr><td>LastSeen</td><td>The time, in UTC, at which the orchestrator last contacted Keyfactor Command.</td></tr>
</table>

Name	Description
Capabilities	

Value	Description
AWS	Amazon Web Services (Deprecated)
AWSCerManA	Amazon Web Services (Suggested Name for Custom GitHub Extension)
CA	Remote CA Management
CitrixAdc	Citrix\NetScaler (Suggested Name for Custom GitHub Extension)
F5-CA-REST	F5 CA Bundles (REST)
F5-WS-REST	F5 Web Server (REST)
F5-SL-REST	F5 SSL Profile (REST)
FTP	File Transfer Protocol (Deprecated)
F5	F5 SSL Profile and F5 Web Server (SOAP) (Deprecated)
IIS	IIS (Deprecated)
IISU	IIS Bound Certificate (Suggested Name for Custom GitHub Extension)
JKS	Java Keystore
LOGS	Fetch Logs
MacEnrollment	Mac Autoenrollment
NS	NetScaler (Deprecated)
PEM	PEM Store
RFJKS	Java Keystore (Suggested Name for Custom GitHub Extension)
RFPkcs12	PKCS#12 Store (Suggested Name for Custom GitHub Extension)
RFPEM	PEM Store (Suggested Name for Custom GitHub Extension)
SSL	SSL Discovery and Monitoring
TemplateSync	Template Synchronization

Name	Description
Blueprint	A string indicating the name of the blueprint associated with the orchestrator.
Thumbprint	A string indicating the thumbprint of the certificate that Keyfactor Command is expecting the orchestrator to use for client certificate authentication.
LegacyThumbprint	A string indicating the thumbprint of the certificate previously used by the orchestrator for client certificate authentication before a certificate renewal operation took place (rotating the current thumbprint into the legacy thumbprint). The legacy thumbprint is cleared once the orchestrator successfully registers with the new thumbprint.
AuthCertificateReenrollment	An integer indicating the value of the orchestrator certificate reenrollment request or require status. Possible values are: **Value** — **Description** 0 — None—Unset the value so that the orchestrator will not request a new client authentication certificate (based on this value). 1 — Requested—The orchestrator will request a new client authentication certificate when it next registers for a session. Orchestrator activity will be allowed to continue as usual. 2 — Required—The orchestrator will request a new client authentication certificate when it next registers for a session. A new session will not be granted and orchestrator activity will not be allowed to continue until the orchestrator acquires a new certificate.
LastThumbprintUsed	A string indicating the thumbprint of the certificate that the orchestrator most recently used for client certificate authentication. In most cases, this will match the *Thumbprint*.
LastErrorCode	An integer indicating the last error code, if any, reported from the orchestrator when trying to register a session. This code is cleared on successful session registration.
LastErrorMessage	A string indicating the last error message, if any, reported from the orchestrator when trying to register a session. This message is cleared on successful session registration.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.1.2 GET Agents

The GET /Agents method is used to retrieve a list of orchestrators and agents registered in Keyfactor Command. This method returns HTTP 200 OK on a success with a list of all orchestrator details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 96: GET Agents Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Orchestrator Management Search Feature on page 507. The query fields supported for this endpoint are: • AgentId • Blueprint • Capabilities (See Table 97: GET Agent Response Data Capabilities) • ClientMachine • ErrorCode • ErrorMessage (last error message) • Identity (Username) • LastSeen (DateTime) • Platform (Platform types: 0-Unknown, 1-.NET, 2-Java, 3-Mac, 4-Android, 5-Native, 6-Bash, 7-Universal Orchestrator) • Status (1-New, 2-Approved, 3-Disapproved) • Version **Tip:** Use the following query to return only approved orchestrators: `Status -eq "2"` A value of 1 will return orchestrators with a status of *New* and a value of 3 will return orchestrators with a status of *Disapproved*.
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.

Name	In	Description
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *AgentId*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 97: GET Agent Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>AgentId</td><td>A string indicating the GUID of the orchestrator.</td></tr>
<tr><td>ClientMachine</td><td>A string indicating the client machine on which the orchestrator is installed.</td></tr>
<tr><td>Username</td><td>A string indicating the Active Directory user or service account the orchestrator is using to connect to Keyfactor Command.</td></tr>
<tr><td>AgentPlatform</td><td>An integer indicating the platform for the orchestrator. Possible values are:
<table>
<tr><th>Value</th><th>Parameter Value</th></tr>
<tr><td>0</td><td>Unknown</td></tr>
<tr><td>1</td><td>Keyfactor Windows Orchestrator</td></tr>
<tr><td>2</td><td>Keyfactor Java Agent</td></tr>
<tr><td>3</td><td>Keyfactor Mac Auto-Enrollment Agent</td></tr>
<tr><td>4</td><td>Keyfactor Android Agent</td></tr>
<tr><td>5</td><td>Keyfactor Native Agent</td></tr>
<tr><td>6</td><td>Keyfactor Bash Orchestrator</td></tr>
<tr><td>7</td><td>Keyfactor Universal Orchestrator</td></tr>
</table></td></tr>
<tr><td>Version</td><td>A string indicating the version of the orchestrator.</td></tr>
<tr><td>Status</td><td>An integer indicating the orchestrator status. Possible values are:
<table>
<tr><th>Value</th><th>Parameter Value</th></tr>
<tr><td>1</td><td>New</td></tr>
<tr><td>2</td><td>Approved</td></tr>
<tr><td>3</td><td>Disapproved</td></tr>
</table></td></tr>
<tr><td>LastSeen</td><td>The time, in UTC, at which the orchestrator last contacted Keyfactor Command.</td></tr>
</table>

Name	Description
Capabilities	

Value	Description
AWS	Amazon Web Services (Deprecated)
AWSCerManA	Amazon Web Services (Suggested Name for Custom GitHub Extension)
CA	Remote CA Management
CitrixAdc	Citrix\NetScaler (Suggested Name for Custom GitHub Extension)
F5-CA-REST	F5 CA Bundles (REST)
F5-WS-REST	F5 Web Server (REST)
F5-SL-REST	F5 SSL Profile (REST)
FTP	File Transfer Protocol (Deprecated)
F5	F5 SSL Profile and F5 Web Server (SOAP) (Deprecated)
IIS	IIS (Deprecated)
IISU	IIS Bound Certificate (Suggested Name for Custom GitHub Extension)
JKS	Java Keystore
LOGS	Fetch Logs
MacEnrollment	Mac Autoenrollment
NS	NetScaler (Deprecated)
PEM	PEM Store
RFJKS	Java Keystore (Suggested Name for Custom GitHub Extension)
RFPkcs12	PKCS#12 Store (Suggested Name for Custom GitHub Extension)
RFPEM	PEM Store (Suggested Name for Custom GitHub Extension)
SSL	SSL Discovery and Monitoring
TemplateSync	Template Synchronization

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Blueprint</td><td>A string indicating the name of the blueprint associated with the orchestrator.</td></tr>
<tr><td>Thumbprint</td><td>A string indicating the thumbprint of the certificate that Keyfactor Command is expecting the orchestrator to use for client certificate authentication.</td></tr>
<tr><td>LegacyThumbprint</td><td>A string indicating the thumbprint of the certificate previously used by the orchestrator for client certificate authentication before a certificate renewal operation took place (rotating the current thumbprint into the legacy thumbprint). The legacy thumbprint is cleared once the orchestrator successfully registers with the new thumbprint.</td></tr>
<tr><td>AuthCertificateReenrollment</td><td>An integer indicating the value of the orchestrator certificate reenrollment request or require status. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>None—Unset the value so that the orchestrator will not request a new client authentication certificate (based on this value).</td></tr><tr><td>1</td><td>Requested—The orchestrator will request a new client authentication certificate when it next registers for a session. Orchestrator activity will be allowed to continue as usual.</td></tr><tr><td>2</td><td>Required—The orchestrator will request a new client authentication certificate when it next registers for a session. A new session will not be granted and orchestrator activity will not be allowed to continue until the orchestrator acquires a new certificate.</td></tr></table></td></tr>
<tr><td>LastThumbprintUsed</td><td>A string indicating the thumbprint of the certificate that the orchestrator most recently used for client certificate authentication. In most cases, this will match the Thumbprint.</td></tr>
<tr><td>LastErrorCode</td><td>An integer indicating the last error code, if any, reported from the orchestrator when trying to register a session. This code is cleared on successful session registration.</td></tr>
<tr><td>LastErrorMessage</td><td>A string indicating the last error message, if any, reported from the orchestrator when trying to register a session. This message is cleared on successful session registration.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.1.3 POST Agents Reset

The POST /Agents/Reset method is used to reset one or more orchestrators, including:

- Remove all current orchestrator jobs for the selected orchestrator(s).
- Delete all associated certificate stores.
- Set the orchestrator status to new.
- For orchestrators configured to use client certificate authentication, clear the certificate thumbprints stored for the orchestrator(s) to allow them to be reconfigured with a new certificate.

This endpoint returns 204 with no content upon success. On a failure, a 400 is returned with an error message.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 98: POST Agents Reset Input Parameters

Name	In	Description
agentIds	Body	**Required**. An array of strings indicating the Keyfactor Command reference GUIDs of the orchestrators to reset. Use the *GET /Agents* method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.1.4 POST Agents Approve

The POST /Agents/Approve method is used to approve one or more orchestrators (a.k.a. agents). An orchestrator must be approved before jobs for it can be scheduled or carried out. This endpoint

returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 99: POST Agents Approve Input Parameters

Name	In	Description
agentIds	Body	**Required**. An array of strings indicating the GUIDs of the orchestrators to approve. Use the *GET Agents* method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs and current status of the orchestrators.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.1.5 POST Agents Disapprove

The POST /Agents/Disapprove method is used to disapprove one or more orchestrators (a.k.a. agents). When an orchestrator is disapproved, operations with Keyfactor Command can no longer be carried out by this orchestrator. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 100: POST Agents Disapprove Input Parameters

Name	In	Description
agentIds	Body	**Required**. An array of strings indicating the orchestrator GUIDs to disapprove. Use the *GET Agents* method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs and current status of the orchestrators.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.1.6 POST Agents ID Reset

The POST /Agents/{id}/Reset method is used to reset a single orchestrator, including:

- Remove all current orchestrator jobs for the selected orchestrator.
- Delete all associated certificate stores.
- Set the orchestrator status to new.
- For orchestrators configured to use client certificate authentication, clear the certificate thumbprints stored for the orchestrator to allow it to be reconfigured with a new certificate.

This endpoint returns 204 with no content upon success. On a failure, a 400 is returned with an error message.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 101: POST Agents {id} Reset Input Parameters

Name	In	Description
id	Path	**Required**. The GUID of the orchestrator to reset. Use the *GET /Agents* method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.1.7 POST Agents ID FetchLogs

The POST /Agents/{id}/FetchLogs method is used to schedule a job on a Native Agent to retrieve log files. The job will be scheduled to run immediately, which means it should complete within a few

minutes depending on other activity occurring at the same time. This method is currently only supported for the Native Agent. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Tip: To schedule a job to retrieve logs from a Keyfactor Universal Orchestrator, use the POST /OrchestratorJobs/Custom method (see POST Orchestrator Jobs Custom on page 1854).

Table 102: POST Agents {id} FetchLogs Input Parameters

Name	In	Description
id	Path	**Required**. The GUID of the orchestrator to schedule the job for. Use the *GET /Agents* method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.1.8 POST Agents Set Auth Certificate Reenrollment

The POST /Agents/SetAuthCertificateReenrollment method is used to request or require that one or more orchestrators (a.k.a. agents) enroll for a new client authentication certificate on the orchestrator's next session registration. This method returns HTTP 200 OK on a success with information about any failed requests.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 103: POST Agents Set Auth Certificate Reenrollment Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>OrchestratorIds</td><td>Body</td><td>Required. An array of strings indicating the GUIDs of the orchestrators on which you want to change the AuthCertificateReenrollment value to request or require the orchestrator(s) to enroll for a new client authentication certificate on the next session registration.
Use the GET Agents method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs and current status of the orchestrators.</td></tr>
<tr><td>Status</td><td>Body</td><td>An integer indicating the value that AuthCertificateReenrollment should be set to. Status options are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>None—Unset the value so that the orchestrator will not request a new client authentication certificate (based on this value).</td></tr><tr><td>1</td><td>Requested—The orchestrator will request a new client authentication certificate when it next registers for a session. Orchestrator activity will be allowed to continue as usual.</td></tr><tr><td>2</td><td>Required—The orchestrator will request a new client authentication certificate when it next registers for a session. A new session will not be granted and orchestrator activity will not be allowed to continue until the orchestrator acquires a new certificate.</td></tr></table></td></tr>
</table>

Table 104: POST Agents Set Auth Certificate Reenrollment Response Data

Name	Description
FailedOrchestratorIds	An array of strings indicating the GUIDs of orchestrators that failed to update.
Status	A string indicating the value for AuthCertificateReenrollment that was requested. Status options are: **Value** / **Description** 0 — None—Unset the value so that the orchestrator will not request a new client authentication certificate (based on this value). 1 — Requested—The orchestrator will request a new client authentication certificate when it next registers for a session. Orchestrator activity will be allowed to continue as usual. 2 — Required—The orchestrator will request a new client authentication certificate when it next registers for a session. A new session will not be granted and orchestrator activity will not be allowed to continue until the orchestrator acquires a new certificate.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.2 Agent BluePrint

The Agent BluePrint component of the Keyfactor API includes methods necessary to list, generate, and apply orchestrator and orchestrator blueprints for orchestrators and agents that support blueprint functionality.

Table 105: Agent BluePrint Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the orchestrator blueprint with the specified GUID.	DELETE Agent BluePrint ID on the

Endpoint	Method	Description	Link
			next page
/{id}	GET	Returns details for the orchestrator blueprint with the specified GUID.	GET Agent BluePrint ID on the next page
/	GET	Returns details for all orchestrator blueprints.	GET Agent BluePrint on page 872
/{id}/Jobs	GET	Returns details of the certificate store scheduled jobs for the orchestrator blueprint with the specified GUID.	GET Agent BluePrint ID Jobs on page 873
/{id}/Stores	GET	Returns details of the certificate stores for the orchestrator blueprint with the specified GUID.	GET Agent BluePrint ID Stores on page 878
/ApplyBlueprint	POST	Applies an orchestrator blueprint to one or more orchestrators.	POST AgentBluePrint ApplyBluePrint on page 881
/GenerateBlueprint	POST	Creates a new orchestrator blueprint from an orchestrator.	POST AgentBluePrint GenerateBluePrint on page 882

3.6.2.1 DELETE Agent BluePrint ID

The DELETE /AgentBluePrint/{id} method is used to delete an existing orchestrator blueprint with the specified blueprint GUID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 106: DELETE Agent BluePrint {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the orchestrator blueprint that should be deleted. Use the *GET AgentBluePrint* method (see GET Agent BluePrint on page 872) to retrieve a list of all the blueprints to determine the orchestrator blueprint GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.2.2 GET Agent BluePrint ID

The GET /AgentBluePrint/{id} method is used to retrieve information about the orchestrator blueprint with the specified blueprint GUID. This method returns HTTP 200 OK on a success with information about the blueprint.

Tip: To see the certificate stores or scheduled jobs associated with the blueprint, use the GET /AgentBluePrint/{id}/Jobs method (see GET Agent BluePrint ID Jobs on page 873) or GET /AgentBluePrint/{id}/Stores method (see GET Agent BluePrint ID Stores on page 878).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 107: GET Agent BluePrint {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the orchestrator blueprint that should be retrieved. Use the *GET AgentBluePrint* method (see GET Agent BluePrint on the next page) to retrieve a list of all the blueprints to determine the orchestrator blueprint GUID.

Table 108: GET Agent BluePrint {id} Response Data

Name	Description
AgentBlueprintId	A string indicating the GUID of the blueprint.
Name	A string indicating the name of the blueprint.
RequiredCapabilities	An array of strings indicating the type of capabilities required by the orchestrators to which the blueprint will be applied (e.g. JKS, PEM).
LastModified	A string indicating the date and time the blueprint was created.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.2.3 GET Agent BluePrint

The GET /AgentBluePrint method is used to retrieve a list of blueprints defined for the orchestrators and agents registered in Keyfactor Command. This method returns HTTP 200 OK on a success with a list of all blueprint details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 109: GET Agent BluePrint Input Parameters

Name	In	Description
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 110: GET Agent BluePrint Response Data

Name	Description
AgentBlueprintId	A string indicating the GUID of the blueprint.
Name	A string indicating the name of the blueprint.
RequiredCapabilities	An array of strings indicating the type of capabilities required by the orchestrators to which the blueprint will be applied (e.g. JKS, PEM).
LastModified	A string indicating the date and time the blueprint was created.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.2.4 GET Agent BluePrint ID Jobs

The GET /AgentBluePrint/{id}/Jobs method is used to retrieve details of the scheduled certificate store jobs for the orchestrator blueprint with the specified blueprint GUID. This method returns HTTP 200 OK on a success with a list of all the blueprint scheduled job details, including certificate stores.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 111: GET Agent BluePrint {id} Jobs Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the orchestrator blueprint that should be retrieved. Use the *GET AgentBluePrint* method (see GET Agent BluePrint on page 872) to retrieve a list of all the blueprints to determine the orchestrator blueprint GUID.
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *StorePath*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 112: GET Agent BluePrint {id} Jobs Response Data

Name	Description
AgentBlueprintJobId	A string indicating the GUID of the certificate store job associated with the blueprint.
AgentBlueprintStoreId	A string indicating the GUID of the certificate store associated with the blueprint.
AgentBlueprintId	A string indicating the GUID of the blueprint.
JobType	A string indicating the GUID of the certificate store job type.
JobTypeName	A string indicating the certificate store job type (e.g. JksInventory).
OperationType	In integer indicating the type of operation (e.g. 2 = add to certificate store, 3 = remove from certificate store).
Thumbprint	A string indicating the thumbprint of the certificate to add to or remove from the certificate store. This field is populated only for management jobs.
Contents	A string containing the certificate to be added to the certificate store. This field is populated only for management add to certificate store jobs.
Alias	A string indicating the alias to be used for the certificate upon entry into or removal from the certificate store. The function of the alias varies depending on the certificate store type. For example, for a Java keystore, it is user-generated and stored in the keystore associated with the certificate while for PEM stores it is the thumbprint of the certificate. Some certificate store types don't require an alias and some do. See Add Certificate on page 74 in the *Keyfactor Command Reference Guide* for more information. This field is populated only for management jobs.
PrivateKeyEntry	A Boolean indicating whether the certificate store has a separate private key file. This field is populated only for management jobs.
Overwrite	A Boolean indicating whether the certificate already in the certificate store should be overwritten with the new certificate, if applicable. This field is populated only for management jobs.
HasEntryPassword	A Boolean indicating whether the certificate in the certificate store has a different password from the certificate store itself. This field is populated only for management jobs.
HasPfxPassword	A Boolean indicating whether the certificate being added to the certificate store has a private key. This field is populated only for management jobs.
RequestTimestamp	A string indicating the time at which the management job was requested. This

Name	Description
	field is populated only for management jobs.
KeyfactorSchedule	An object containing the schedule for the certificate store job. This field is populated only for inventory and discovery jobs.
Subject	A string containing the reenrollment subject name using X.500 format. This field is populated only for reenrollment jobs.
Directories	A string containing the directory or directories to search during a discovery job. This field is populated only for discovery jobs.
IgnoredDirectories	A string containing the directories that should not be included in the search during discovery jobs. This field is populated only for discovery jobs.
SymLinks	A Boolean indicating whether the job should follow symbolic links on Linux and UNIX operating systems and report both the actual location of a found certificate store file in addition to the symbolic link pointing to the file during discovery jobs. This option is ignored on Windows. This field is populated only for discovery jobs.
Compatibility	A Boolean indicating whether the job will run using the compatibility mode introduced in Java version 1.8 to locate both JKS and PKCS12 type files (true) or not (false) during Java keystore discovery jobs. This field is populated only for discovery jobs.
FileExtensions	A string containing the file extensions for which to search during a discovery job. For example, search for files with the extension “jks” in order to exclude files with other extensions such as “txt”. This field is populated only for discovery jobs.
FileNamePatterns	A string against which to compare the file names of certificate store files and return only those that contain the specified string (e.g. myjks) during discovery jobs. This field is populated only for discovery jobs.
AgentBlueprintStores	An object that includes the certificate store information of the job. The following certificate store details are included:

Name	Description
AgentBlueprintStoreId	A string indicating the GUID of the certificate store associated with the blueprint.
AgnetBlueprintId	A string indicating the GUID of the blueprint.

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>StorePath</td><td>A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks). See Adding or Modifying a Certificate Store on page 413 in the *Keyfactor Command Reference Guide* for more information.</td></tr><tr><td>ContainerId</td><td>An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).</td></tr><tr><td>CertStoreType</td><td>An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.</td></tr><tr><td>CertStoreTypeName</td><td>A string indicating a reference name for the certificate store type (e.g. Java Keystore, PEM File).</td></tr><tr><td>Approved</td><td>A Boolean indicating whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.</td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>CreateIfMissing</td><td>A Boolean indicating whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.</td></tr><tr><td>Properties</td><td>A string containing additional properties for the store. Only some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information).</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.2.5 GET Agent BluePrint ID Stores

The GET /AgentBluePrint/{id}/Stores method is used to retrieve details of the certificate stores for the orchestrator blueprint with the specified blueprint GUID. This method returns HTTP 200 OK on a success with a list of all the blueprint certificate store details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 113: GET Agent BluePrint {id} Stores Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the orchestrator blueprint that should be retrieved. Use the *GET AgentBluePrint* method (see GET Agent BluePrint on page 872) to retrieve a list of all the blueprints to determine the orchestrator blueprint GUID.
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *StorePath*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 114: GET Agent BluePrint {id} Stores Response Data

Name	Description
AgentBlueprintStoreId	A string indicating the GUID of the certificate store associated with the blueprint.
AgentBlueprintId	A string indicating the GUID of the blueprint.
StorePath	A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks). See Adding or Modifying a Certificate Store on page 413 in the *Keyfactor Command Reference Guide* for more information.
ContainerId	An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
CertStoreTypeName	A string indicating a reference name for the certificate store type (e.g. Java Keystore, PEM File).
Approved	A Boolean indicating whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.
CreateIfMissing	A Boolean indicating whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.
Properties	Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.2.6 POST AgentBluePrint ApplyBluePrint

The POST /AgentBluePrint/ApplyBluePrint method is used to apply a blueprint with associated certificate stores and scheduled jobs to an orchestrator. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 115: POST Agent BluePrint Apply BluePrint Input Parameters

Name	In	Description
templateId	Query	A string indicating the Keyfactor Command GUID of the blueprint to apply to the orchestrator(s). Use the *GET AgentBluePrint* method (see GET Agent BluePrint on page 872) to retrieve a list of all the blueprints to determine the blueprint GUIDs.
	Query	**Required**. An array of strings indicating the GUIDs of the orchestrators to which the blueprint should be applied. Use the *GET Agents* method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs and current status of the orchestrators. **Note:** Orchestrators must be approved before a blueprint can be applied.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.2.7 POST AgentBluePrint GenerateBluePrint

The POST /AgentBluePrint/GenerateBluePrint method is used to create a new blueprint based on the certificate stores and scheduled jobs of one orchestrator. This method returns HTTP 200 OK on a success with details of the new blueprint.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 116: POST Agent BluePrint Generate Input Parameters

Name	In	Description
agentIds	Body	**Required**. A string indicating the GUID of the orchestrator that should be used to generate the blueprint. Use the *GET Agents* method (see GET Agents on page 858) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs and current status of the orchestrators.
name	Body	**Required**. A string indicating the name for the new blueprint.

Table 117: POST Agent BluePrint Generate Response Data

Name	Description
AgentBlueprintId	A string indicating the GUID of the blueprint.
Name	A string indicating the name of the blueprint.
RequiredCapabilities	An array of strings indicating the type of capabilities required by the orchestrators to which the blueprint will be applied (e.g. JKS, PEM).
lastModified	A string indicating the date the blueprint was generated in UTC time.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.3 Agent Pools

The Agent Pools component of the Keyfactor API includes methods necessary to programmatically add, edit, get, and delete Agent Pools. An orchestrator (a.k.a. agent) pool is a group of Keyfactor Command Windows Orchestrators and/or Universal Orchestrators that have the SSL capability. Each pool is used to divide the work of scanning a network between all orchestrators that are members of it.

Table 118: Agent Pool Endpoints

Endpoint	Method	Description	Links
/{id}	DELETE	Deletes the specified orchestrator pool.	DELETE Agent Pools ID below
/{id}	GET	Returns limited information about the orchestrators in the specified pool.	GET Agent Pools ID on the next page
/	GET	Returns a list of all orchestrator pools with limited information about the orchestrators assigned to each pool.	GET Agent Pools on page 886
/	POST	Creates an orchestrator pool based on information in the request.	POST Agent Pools on page 889
/	PUT	Updates an orchestrator pool based on information in the request.	PUT Agent Pools on page 891
/Agents	GET	Returns a list of orchestrators associated with the Default Agent Pool.	GET Agent Pools Agents on page 894

3.6.3.1 DELETE Agent Pools ID

The DELETE /AgentPools/{id} method is used to delete an existing orchestrator (a.k.a. agent) pool. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/
/ssl/modify/

Table 119: DELETE Agent Pools {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the orchestrator pool to delete. Use the *GET /AgentPools* method (see GET Agent Pools on page 886) to retrieve a list of all the orchestrator pools to determine the orchestrator pool GUID. The Default Agent Pool cannot be deleted.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.3.2 GET Agent Pools ID

The GET /AgentPools/{id} method is used to return information about a single orchestrator (a.k.a. agent) pool. This method returns HTTP 200 OK on a success with details about the requested orchestrator pool.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 120: GET Agent Pools {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the orchestrator pool to retrieve. Use the *GET /AgentPools* method (see GET Agent Pools on page 886) to retrieve a list of all the orchestrator pools to determine the orchestrator pool GUID.

Table 121: GET AgentPools {id} Response Data

Name	Description
AgentPoolId	A string indicating the GUID of the orchestrator pool.
Name	A string indicating the name of the orchestrator pool.
DiscoverAgentsCount	An integer specifying the number of orchestrators in the pool that can perform discovery jobs.
MonitorAgentsCount	An integer specifying the number of orchestrators in the pool that can perform monitoring jobs.
Agents	An array of objects containing the orchestrators that are assigned to the orchestrator pool, with accompanying data about the orchestrators. Orchestrator details include:

Name	Description
AgentId	A string indicating the GUID of the orchestrator.
EnableDiscover	A Boolean that indicates whether this orchestrator is allowed to perform discovery jobs for the orchestrator pool to which it has been assigned (true) or not (false).
EnableMonitor	A Boolean that indicates whether this orchestrator is allowed to perform monitoring jobs for the orchestrator pool to which it has been assigned (true) or not (false).
Version	A string indicating the version of the orchestrator.
AllowsDiscover	A Boolean that indicates whether this orchestrator has the capability to perform discovery jobs (true) or not (false).
AllowsMonitor	A Boolean that indicates whether this orchestrator has the capability to perform monitoring jobs (true) or not (false).
ClientMachine	A string indicating the client machine on which the orchestrator is installed.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.3.3 GET Agent Pools

The GET /AgentPools method is used to retrieve all orchestrator (a.k.a. agent) pools. This method returns HTTP 200 OK on a success with a list of all agent pool details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 122: GET Agent Pools Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • *Id* (AgentPoolID) • *Name*
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 123: GET AgentPools Response Data

Name	Description
AgentPoolId	A string indicating the GUID of the orchestrator pool.
Name	A string indicating the name of the orchestrator pool.
DiscoverAgentsCount	An integer specifying the number of orchestrators in the pool that can perform discovery jobs.
MonitorAgentsCount	An integer specifying the number of orchestrators in the pool that can perform monitoring jobs.
Agents	An array of objects containing the orchestrators that are assigned to the orchestrator pool, with accompanying data about the orchestrators. Orchestrator details include:

Name	Description
AgentId	A string indicating the GUID of the orchestrator.
EnableDiscover	A Boolean that indicates whether this orchestrator is allowed to perform discovery jobs for the orchestrator pool to which it has been assigned (true) or not (false).
EnableMonitor	A Boolean that indicates whether this orchestrator is allowed to perform monitoring jobs for the orchestrator pool to which it has been assigned (true) or not (false).
Version	A string indicating the version of the orchestrator.
AllowsDiscover	A Boolean that indicates whether this orchestrator has the capability to perform discovery jobs (true) or not (false).
AllowsMonitor	A Boolean that indicates whether this orchestrator has the capability to perform monitoring jobs (true) or not (false).
ClientMachine	A string indicating the client machine on which the orchestrator is installed.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.3.4 POST Agent Pools

The POST /AgentPools method is used to create a new orchestrator (a.k.a. agent) pool. This method returns HTTP 200 OK on a success with information about the orchestrator pool.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/
/ssl/modify/

Table 124: POST Agent Pools Input Parameters

Name	In	Description
Name	Body	**Required**. A string indicating the name of the orchestrator pool.
Agents	Body	A list of orchestrators that will be part of this orchestrator pool. The orchestrators **must not be** assigned to a different orchestrator pool (except the Default Agent Pool). Per orchestrator data that can be provided includes: **Name** / **Description** AgentId — **Required**. A string indicating the GUID of the orchestrator being assigned. EnableDiscover — **Required***. A Boolean that sets whether a discovery job can be sent to this orchestrator. One of *EnabledDiscover* or *EnableMonitor* is **required**. EnableMonitor — **Required***. A Boolean that sets whether a monitoring job can be sent to this orchestrator. One of *EnabledDiscover* or *EnableMonitor* is **required**.

Table 125: POST Agent Pools Response Data

Name	Description
AgentPoolId	A string indicating the GUID of the orchestrator pool.
Name	A string indicating the name of the orchestrator pool.
DiscoverAgentsCount	An integer specifying the number of orchestrators in the pool that can perform discovery jobs.
MonitorAgentsCount	An integer specifying the number of orchestrators in the pool that can perform monitoring jobs.
Agents	An array of objects containing the orchestrators that are assigned to the orchestrator pool, with accompanying data about the orchestrators. Orchestrator details include:

Name	Description
AgentId	A string indicating the GUID of the orchestrator.
EnableDiscover	A Boolean that indicates whether this orchestrator is allowed to perform discovery jobs for the orchestrator pool to which it has been assigned (true) or not (false).
EnableMonitor	A Boolean that indicates whether this orchestrator is allowed to perform monitoring jobs for the orchestrator pool to which it has been assigned (true) or not (false).
Version	A string indicating the version of the orchestrator.
AllowsDiscover	A Boolean that indicates whether this orchestrator has the capability to perform discovery jobs (true) or not (false).
AllowsMonitor	A Boolean that indicates whether this orchestrator has the capability to perform monitoring jobs (true) or not (false).
ClientMachine	A string indicating the client machine on which the orchestrator is installed.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.3.5 PUT Agent Pools

The PUT /AgentPools method is used to update an existing orchestrator (a.k.a. agent) pool. This method returns HTTP 200 OK on a success with information about the orchestrator pool.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/
/ssl/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 126: PUT Agent Pools Input Parameters

Name	In	Description
AgentPoolId	Body	**Required**. A string indicating the GUID of the orchestrator pool that is to be updated.
Name	Body	**Required**. A string indicating the name of the orchestrator pool.
Agents	Body	A list of orchestrators that will be part of this orchestrator pool. The orchestrators **must not be** assigned to a different orchestrator pool (except the Default Agent Pool). Per orchestrator data that can be provided includes: **Name** / **Description** AgentId — **Required**. A string indicating the GUID of the orchestrator being assigned. EnableDiscover — **Required***. A Boolean that sets whether a discovery job can be sent to this orchestrator. One of *EnabledDiscover* or *EnableMonitor* is **required**. EnableMonitor — **Required***. A Boolean that sets whether a monitoring job can be sent to this orchestrator. One of *EnabledDiscover* or *EnableMonitor* is **required**.

Table 127: PUT Agent Pools Response Data

Name	Description
AgentPoolId	A string indicating the GUID of the orchestrator pool.
Name	A string indicating the name of the orchestrator pool.
DiscoverAgentsCount	An integer specifying the number of orchestrators in the pool that can perform discovery jobs.
MonitorAgentsCount	An integer specifying the number of orchestrators in the pool that can perform monitoring jobs.
Agents	An array of objects containing the orchestrators that are assigned to the orchestrator pool, with accompanying data about the orchestrators. Orchestrator details include:

Name	Description
AgentId	A string indicating the GUID of the orchestrator.
EnableDiscover	A Boolean that indicates whether this orchestrator is allowed to perform discovery jobs for the orchestrator pool to which it has been assigned (true) or not (false).
EnableMonitor	A Boolean that indicates whether this orchestrator is allowed to perform monitoring jobs for the orchestrator pool to which it has been assigned (true) or not (false).
Version	A string indicating the version of the orchestrator.
AllowsDiscover	A Boolean that indicates whether this orchestrator has the capability to perform discovery jobs (true) or not (false).
AllowsMonitor	A Boolean that indicates whether this orchestrator has the capability to perform monitoring jobs (true) or not (false).
ClientMachine	A string indicating the client machine on which the orchestrator is installed.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.3.6 GET Agent Pools Agents

The GET /AgentPools/Agents method is used to retrieve the orchestrators (a.k.a. agents) associated with the Default Agent Pool. This method has no required input parameters. It returns HTTP 200 OK on a success with information about the Default Agent Pool orchestrators.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 128: GET Agent Pools Default Agent Pool Agents Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Collection Manager on page 85. The query fields supported for this endpoint are: • Id (Orchestrator ID, AgentID) • ClientMachine • EnableDiscover (true or false) • EnableMonitor (true or false) • Version
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *AgentId*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 129: GET Agent Pools Default Agent Pool Agents Response Data

Name	Description
AgentId	A string indicating the GUID of the orchestrator.
EnableDiscover	A Boolean that indicates whether this orchestrator is allowed to perform discovery jobs for the orchestrator pool to which it has been assigned (true) or not (false).
EnableMonitor	A Boolean that indicates whether this orchestrator is allowed to perform monitoring jobs for the orchestrator pool to which it has been assigned (true) or not (false).
Version	A string indicating the version of the orchestrator.
AllowsDiscover	A Boolean that indicates whether this orchestrator has the capability to perform discovery jobs (true) or not (false).
AllowsMonitor	A Boolean that indicates whether this orchestrator has the capability to perform monitoring jobs (true) or not (false).
ClientMachine	A string indicating the client machine on which the orchestrator is installed.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.4 Alerts

The Alerts component of the Keyfactor API includes methods necessary to create, update, retrieve, schedule, test and delete alerts for denied certificate requests, expired certificates, issued certificate requests, pending certificate requests and SSH Key Rotations.

- Alerts Denied below
- Alerts Expiration on page 927
- Alerts Issued on page 966
- Alerts Key Rotation on page 1001
- Alerts Pending on page 1034

3.6.4.1 Alerts Denied

The Alerts Denied component of the Keyfactor API includes methods necessary to create, update, retrieve, and delete alerts for denied certificate requests.

Table 130: Alerts Denied

Endpoint	Method	Description	Link
/Alerts/Denied/{id}	DELETE	Deletes a denied certificate request alert for the specified ID.	DELETE Alerts Denied ID below
/Alerts/Denied/{id}	GET	Retrieves details for a denied certificate request alert for the specified ID.	GET Alerts Denied ID on the next page
/Alerts/Denied	PUT	Updates a denied certificate request alert for the specified ID.	PUT Alerts Denied on page 917
/Alerts/Denied	GET	Retrieves details for all configured denied certificate request alerts.	GET Alerts Denied on page 902
/Alerts/Denied	POST	Creates a new denied certificate request alert.	POST Alerts Denied on page 907

DELETE Alerts Denied ID

The DELTE /Alerts/Denied/{id} method is used to delete the denied certificate request alert with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 131: DELETE Alerts Denied {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the denied certificate request alert to be deleted. Use the *GET /Alerts/Denied* method (see GET Alerts Denied on page 902) to retrieve a list of all the issued request alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Denied ID

The GET /Alerts/Denied/{id} method is used to retrieve details for the denied certificate request alerts with the specified ID. This method returns HTTP 200 OK on a success with details about the specified denied certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 132: GET Alerts Denied {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the denied certificate request alert. Use the *GET /Alerts/Denied* method (see GET Alerts Denied on page 902) to retrieve a list of all the issued request alerts to determine the alert ID.

Table 133: GET Alerts Denied {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the denied request alert.
DisplayName	A string indicating the display name for the denied request alert. This name appears in the denied request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nWe are sorry to report that the certificate you requested on {subdate} in the name {rcn} has not been issued for the following reason:\n\n{cmnt}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Template: {template}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>CA: {careqid}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.

</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the denied request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all denied certificate requests. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for all templates.</td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr>
<tr><td>ForestRoot</td><td>A string indicating the forest root of the template.

Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr>
<tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>6</td><td>DeniedLogger</td></tr>
<tr><td>7</td><td>DeniedPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Denied

The GET /Alerts/Denied method is used to retrieve details of all denied certificate request alerts configured in Keyfactor Command. Results can be limited to selected alerts using filtering, and URL parameters can be used to specify paging and sorting. This method returns HTTP 200 OK on a success with details about the specified denied certificate request alerts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 134: GET Alerts Denied Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • DisplayName • Message • RegisteredEventHandlerId • Subject • Template_Id • UseHandler
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 135: GET Alerts Denied Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the denied request alert.
DisplayName	A string indicating the display name for the denied request alert. This name appears in the denied request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nWe are sorry to report that the certificate you requested on {subdate} in the name {rcn} has not been issued for the following reason:\n\n{cmnt}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Template: {template}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>CA: {careqid}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.

</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the denied request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all denied certificate requests. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for all templates.</td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr>
<tr><td>ForestRoot</td><td>A string indicating the forest root of the template.

Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr>
<tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>6</td><td>DeniedLogger</td></tr>
<tr><td>7</td><td>DeniedPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Denied

The POST /Alerts/Denied method is used to create a new denied certificate request alert. This method returns HTTP 200 OK on a success with details about the denied certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 136: POST Alerts Denied Input Parameters

Name	In	Description
DisplayName	Body	**Required**. A string indicating the display name for the denied request alert. This name appears in the denied request alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nWe are sorry to report that the certificate you requested on {subdate} in the name {rcn} has not been issued for the following reason:\n\n{cmnt}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Template: {template}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>CA: {certqid}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings. **Note:** The $(requester:givenname) substitutable special text token is only supported in environments using Active

Name	In	Description
		Directory as an identity provider.
Recipients	Body	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail} The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.
TemplateId	Body	An integer indicating the certificate template for which the denied request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all denied certificate requests. Use the GET /Templates method (see GET Templates on page 2422) to retrieve a list of all the templates to determine the template ID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>6</td><td>DeniedLogger</td></tr>
<tr><td>7</td><td>DeniedPowershell</td></tr>
</table></td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:</td></tr>
</table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• LogTarget
		This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged.
		• Script
		This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
		• Token
		This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings.
		• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		  {</pre>

Name	In	Description

		"Id": 28,
		"Key": "cn",
		"DefaultValue": "rcn",
		"ParameterType": "Token"
		},
		{
		"Id": 29,
		"Key": "AppOwnerFirstName",
		"DefaultValue": "metadata:AppOwnerFirstName",
		"ParameterType": "Token"
		},
		{
		"Id": 30,
		"Key": "Text",
		"DefaultValue": "Denied Alert: Enterprise Web Server",
		"ParameterType": "Value"
		},
		{
		"Id": 31,
		"Key": "DenialComment",
		"DefaultValue": "cmnt",
		"ParameterType": "Token"
		},
		{
		"Id": 32,
		"Key": "ScriptName",
		"DefaultValue": "MyScript.ps1",
		"ParameterType": "Script"
		}
		]

Table 137: POST Alerts Denied Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the denied request alert.
DisplayName	A string indicating the display name for the denied request alert. This name appears in the denied request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nWe are sorry to report that the certificate you requested on {subdate} in the name {rcn} has not been issued for the following reason:\n\n{cmnt}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Template: {template}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>CA: {careqid}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.

</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the denied request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all denied certificate requests. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for all templates.</td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr>
<tr><td>ForestRoot</td><td>A string indicating the forest root of the template.

Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr>
<tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>6</td><td>DeniedLogger</td></tr>
<tr><td>7</td><td>DeniedPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Denied

The PUT /Alerts/Denied method is used to update a denied certificate request alert. This method returns HTTP 200 OK on a success with details about the denied certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 138: PUT Alerts Denied Input Parameters

Name	In	Description
id	Path	An integer indicating the Keyfactor Command reference ID of the denied request alert.
DisplayName	Body	**Required**. A string indicating the display name for the denied request alert. This name appears in the denied request alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: “Hello {requester:givenname},\n\nWe are sorry to report that the certificate you requested on {subdate} in the name {rcn} has not been issued for the following reason:\n\n{cmnt}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Template: {template}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>CA: {careqid}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System” See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings.

Name	In	Description
		Note: The $(requester:givenname) substitutable special text token is only supported in environments using Active Directory as an identity provider.
Recipients	Body	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail} The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.
TemplateId	Body	An integer indicating the certificate template for which the denied request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all denied certificate requests. Use the GET /Templates method (see GET Templates on page 2422) to retrieve a list of all the templates to determine the template ID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>6</td><td>DeniedLogger</td></tr>
<tr><td>7</td><td>DeniedPowershell</td></tr>
</table></td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:</td></tr>
</table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• LogTarget
		This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged.
		• Script
		This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
		• Token
		This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings.
		• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		  {</pre>

Name	In	Description

		"Id": 28,
		"Key": "cn",
		"DefaultValue": "rcn",
		"ParameterType": "Token"
		},
		{
		"Id": 29,
		"Key": "AppOwnerFirstName",
		"DefaultValue": "metadata:AppOwnerFirstName",
		"ParameterType": "Token"
		},
		{
		"Id": 30,
		"Key": "Text",
		"DefaultValue": "Denied Alert: Enterprise Web Server",
		"ParameterType": "Value"
		},
		{
		"Id": 31,
		"Key": "DenialComment",
		"DefaultValue": "cmnt",
		"ParameterType": "Token"
		},
		{
		"Id": 32,
		"Key": "ScriptName",
		"DefaultValue": "MyScript.ps1",
		"ParameterType": "Script"
		}
		]

Table 139: PUT Alerts Denied Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the denied request alert.
DisplayName	A string indicating the display name for the denied request alert. This name appears in the denied request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nWe are sorry to report that the certificate you requested on {subdate} in the name {rcn} has not been issued for the following reason:\n\n{cmnt}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Template: {template}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>CA: {careqid}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.

</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the denied request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all denied certificate requests. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for all templates.</td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr>
<tr><td>ForestRoot</td><td>A string indicating the forest root of the template.

Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr>
<tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>6</td><td>DeniedLogger</td></tr>
<tr><td>7</td><td>DeniedPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 7: Substitutable Special Text for Denied Certificate Request Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.4.2 Alerts Expiration

The Alerts Expiration component of the Keyfactor API includes methods necessary to create, update, retrieve, schedule, and delete alerts for expired certificates.

Table 140: Alerts Expiration

Endpoint	Method	Description	Link
/Alerts/Expiration/{id}	DELETE	Deletes an expired certificate alert for the specified ID.	DELETE Alerts Expiration ID below
/Alerts/Expiration/{id}	GET	Retrieves details for an expired certificate alert for the specified ID.	GET Alerts Expiration ID on the next page
/Alerts/Expiration/Schedule	GET	Retrieves details of the schedule for delivery of expired certificate alerts.	GET Alerts Expiration Schedule on page 933
/Alerts/Expiration/Schedule	PUT	Updates the schedule for delivery of expired certificate alerts.	PUT Alerts Expiration Schedule on page 934
/Alerts/Expiration	GET	Retrieves details for all configured expired certificate alerts.	GET Alerts Expiration on page 936
/Alerts/Expiration	POST	Creates a new expired certificate alert.	POST Alerts Expiration on page 941
/Alerts/Expiration	PUT	Updates an expired certificate for the specified ID.	PUT Alerts Expiration on page 951
/Alerts/Expiration/Test	POST	Test an Expiration Alert	POST Alerts Expiration Test on page 961
/Alerts/Expiration/TestAll	POST	Test All Expiration Alerts	POST Alerts Expiration Test All on page 964

DELETE Alerts Expiration ID

The DELETE /Alerts/Expiration/{id} method is used to delete the expiration alert with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 141: DELETE Alerts Expiration {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the expiration alert to be deleted. Use the *GET /Alerts/Expiration* method (see GET Alerts Expiration on page 936) to retrieve a list of all the expiration alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Expiration ID

The GET /Alerts/Expiration/{id} method is used to retrieve details for the expiration alert with the specified ID. This method returns HTTP 200 OK on a success with details about the specified alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 142: GET Alerts Expiration {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the expiration alert. Use the *GET /Alerts/Expiration* method (see GET Alerts Expiration on page 936) to retrieve a list of all the expiration alerts to determine the alert ID.

Table 143: GET Alerts Expiration {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the expiration alert.
DisplayName	A string indicating the display name for the expiration alert. This name appears in the Expiration Certificate Request Alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings.
ExpirationWarningDays	An integer indicating the number of days prior to expiration to send the warning. **Example:** When alerts run, the alert engine reports on all the certificates expiring within the next X days (e.g. 30 days) from the execution time that have not previously been reported on. This means that if the alerts run daily and have been running daily regularly for some time, only a single day of expiring certificates will be reported on by any given alert run. For example, say you create a new alert that has never run before for collection A and set it to 30 days. You configure it to run daily at 5:00 am. The alert runs for the first time at 5:00 am on July 1st. All the certificates in collection A that will expire between July 1st at 12:oo am UTC and July 31 at 12:00 am UTC will be alerted on. The next day when the alert runs again at 5:00 am on July 2nd, the certificates in collection A expiring between July 31st at 12:00 am UTC and August 1st at 12:00 am UTC will be alerted on. If alerts are missed for a period of time (due to an outage, for example), the next run of the alerts will check the previous successful run date for the alerts and report on certificates expiring X days from that outage window. For example, using the collection A alert referenced above, say an outage caused the alerts not to run on August 1 and August 2. On August 3, the alert would run again at 5:00 am, and the certificates in collection A expiring between August 30th at 12:00 am UTC and September 2nd at 12:00 am UTC would be alerted on.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Recipients</td><td>An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include:
{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQuery</td><td>An object indicating the certificate collection on which the alert is based. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the certificate collection.</td></tr><tr><td>Name</td><td>A string containing the name of the certificate collection.</td></tr></table>For more information about certificate collections, see Saving Search Criteria as a Collection on page 42.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>1</td><td>ExpirationLogger</td></tr>
<tr><td>2</td><td>ExpirationPowershell</td></tr>
<tr><td>3</td><td>ExpirationRenewal</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• LogTarget This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Expiration Schedule

The GET /Alerts/Expiration/Schedule method is used to retrieve the schedule for delivery of expiration alerts configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the schedule for expiration alerts. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 144: GET Alerts Expiration Schedule Response Data

Name	Description
Schedule	An object indicating the schedule for delivery of the expiration alerts. Possible values are: **Name** / **Description** Daily — A dictionary that indicates a job scheduled to run every day at the same time with the parameter: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2023-11-25T23:30:00Z" }`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Expiration Schedule

The PUT /Alerts/Expiration/Schedule method is used to create or update the schedule for delivery of expiration alerts configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the schedule for the alerts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/monitoring/alerts/modify/

Table 145: PUT Alerts Expiration Schedule Input Parameters

Name	In	Description
Schedule	Body	An object indicating the schedule for delivery of the expiration alerts. Possible values are: **Name** / **Description** Daily — A dictionary that indicates a job scheduled to run every day at the same time with the parameter: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2023-11-25T23:30:00Z" }`

Table 146: PUT Alerts Expiration Schedule Response Data

Name	Description
Schedule	An object indicating the schedule for delivery of the expiration alerts. Possible values are: **Daily** — A dictionary that indicates a job scheduled to run every day at the same time with the parameter: **Time** — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2023-11-25T23:30:00Z" }`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Expiration

The GET /Alerts/Expiration method is used to retrieve details of all expiration alerts configured in Keyfactor Command. Results can be limited to selected alerts using filtering, and URL parameters can be used to specify paging and sorting. This method returns HTTP 200 OK on a success with details about the specified alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 147: GET Alerts Expiration Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • CertificateQueryId • Days • DisplayName • Message • RegisteredEventHandlerId • ScheduledTaskId • Subject • UseHandler
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 148: GET Alerts Expiration Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the expiration alert.
DisplayName	A string indicating the display name for the expiration alert. This name appears in the Expiration Certificate Request Alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings.
ExpirationWarningDays	An integer indicating the number of days prior to expiration to send the warning. **Example:** When alerts run, the alert engine reports on all the certificates expiring within the next X days (e.g. 30 days) from the execution time that have not previously been reported on. This means that if the alerts run daily and have been running daily regularly for some time, only a single day of expiring certificates will be reported on by any given alert run. For example, say you create a new alert that has never run before for collection A and set it to 30 days. You configure it to run daily at 5:00 am. The alert runs for the first time at 5:00 am on July 1st. All the certificates in collection A that will expire between July 1st at 12:oo am UTC and July 31 at 12:00 am UTC will be alerted on. The next day when the alert runs again at 5:00 am on July 2nd, the certificates in collection A expiring between July 31st at 12:00 am UTC and August 1st at 12:00 am UTC will be alerted on. If alerts are missed for a period of time (due to an outage, for example), the next run of the alerts will check the previous successful run date for the alerts and report on certificates expiring X days from that outage window. For example, using the collection A alert referenced above, say an outage caused the alerts not to run on August 1 and August 2. On August 3, the alert would run again at 5:00 am, and the certificates in collection A expiring between August 30th at 12:00 am UTC and September 2nd at 12:00 am UTC would be alerted on.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Recipients</td><td>An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include:
{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQuery</td><td>An object indicating the certificate collection on which the alert is based. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the certificate collection.</td></tr><tr><td>Name</td><td>A string containing the name of the certificate collection.</td></tr></table>For more information about certificate collections, see Saving Search Criteria as a Collection on page 42.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>1</td><td>ExpirationLogger</td></tr>
<tr><td>2</td><td>ExpirationPowershell</td></tr>
<tr><td>3</td><td>ExpirationRenewal</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• LogTarget This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Expiration

The POST /Alerts/Expiration method is used to create a new expiration alert. This method returns HTTP 200 OK on a success with details about the expiration alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 149: POST Alerts Expiration Input Parameters

Name	In	Description
DisplayName	Body	**Required**. A string indicating the display name for the expiration alert. This name appears in the Expiration Certificate Request Alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {cn} in the alert definition and each alert generated at processing time will contain the specific common name of the given certificate instead of the variable {cn}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nThe certificate in the name {cn} issued on {certnotbefore} from {CAreqID} using the {template} template will expire on {certnotafter}. If this certificate is still in use, please consider getting a new one.\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Serial Number: {serial}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>Thumbprint: {thumbprint}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings. **Note:** The $(requester:givenname) substitutable special text token is only supported in environments using Active Directory as an identity provider.

Name	In	Description
ExpirationWarningDays	Body	**Required**. An integer indicating the number of days prior to expiration to send the warning. **Example:** When alerts run, the alert engine reports on all the certificates expiring within the next X days (e.g. 30 days) from the execution time that have not previously been reported on. This means that if the alerts run daily and have been running daily regularly for some time, only a single day of expiring certificates will be reported on by any given alert run. For example, say you create a new alert that has never run before for collection A and set it to 30 days. You configure it to run daily at 5:00 am. The alert runs for the first time at 5:00 am on July 1st. All the certificates in collection A that will expire between July 1st at 12:oo am UTC and July 31 at 12:00 am UTC will be alerted on. The next day when the alert runs again at 5:00 am on July 2nd, the certificates in collection A expiring between July 31st at 12:00 am UTC and August 1st at 12:00 am UTC will be alerted on. If alerts are missed for a period of time (due to an outage, for example), the next run of the alerts will check the previous successful run date for the alerts and report on certificates expiring X days from that outage window. For example, using the collection A alert referenced above, say an outage caused the alerts not to run on August 1 and August 2. On August 3, the alert would run again at 5:00 am, and the certificates in collection A expiring between August 30th at 12:00 am UTC and September 2nd at 12:00 am UTC would be alerted on.
Recipients	Body	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail} The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>Directory as an identity provider.

• Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQueryId</td><td>Body</td><td>Required. An integer indicating the certificate collection on which to base the alert.
Use the GET /CertificateCollections method (see <a>GET Certificate Collections on page 1296) to retrieve a list of all the certificate collections to determine the collection ID.</td></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.

<table><tr><th>ID</th><th>Event Handler Type</th></tr><tr><td>1</td><td>ExpirationLogger</td></tr><tr><td>2</td><td>ExpirationPowershell</td></tr><tr><td>3</td><td>ExpirationRenewal</td></tr></table></td></tr><tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr></table>
For more information about event handlers, see <a>Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:</td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr><tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr><tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see *ParameterType*).</td></tr><tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are: - LogTarget This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. - Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). - Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings. </td></tr></table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		  {
		  "Id": 28,
		  "Key": "cn",
		  "DefaultValue": "cn",
		  "ParameterType": "Token"
		  },
		  {
		  "Id": 29,
		  "Key": "AppOwnerFirstName",
		  "DefaultValue": "metadata:AppOwnerFirstName",
		  "ParameterType": "Token"
		  },
		  {
		  "Id": 30,
		  "Key": "Text",
		  "DefaultValue": "Expiration Alert: Enterprise Web Server",
		  "ParameterType": "Value"
		  },
		  {
		  "Id": 32,
		  "Key": "ScriptName",
		  "DefaultValue": "MyScript.ps1",
		  "ParameterType": "Script"
		  }
		]</pre>

Table 150: POST Alerts Expiration Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the expiration alert.
DisplayName	A string indicating the display name for the expiration alert. This name appears in the Expiration Certificate Request Alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings.
ExpirationWarningDays	An integer indicating the number of days prior to expiration to send the warning. **Example:** When alerts run, the alert engine reports on all the certificates expiring within the next X days (e.g. 30 days) from the execution time that have not previously been reported on. This means that if the alerts run daily and have been running daily regularly for some time, only a single day of expiring certificates will be reported on by any given alert run. For example, say you create a new alert that has never run before for collection A and set it to 30 days. You configure it to run daily at 5:00 am. The alert runs for the first time at 5:00 am on July 1st. All the certificates in collection A that will expire between July 1st at 12:oo am UTC and July 31 at 12:00 am UTC will be alerted on. The next day when the alert runs again at 5:00 am on July 2nd, the certificates in collection A expiring between July 31st at 12:00 am UTC and August 1st at 12:00 am UTC will be alerted on. If alerts are missed for a period of time (due to an outage, for example), the next run of the alerts will check the previous successful run date for the alerts and report on certificates expiring X days from that outage window. For example, using the collection A alert referenced above, say an outage caused the alerts not to run on August 1 and August 2. On August 3, the alert would run again at 5:00 am, and the certificates in collection A expiring between August 30th at 12:00 am UTC and September 2nd at 12:00 am UTC would be alerted on.

Name	Description
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail} The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. • Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.
CertificateQuery	An object indicating the certificate collection on which the alert is based. Possible values are: **Value** / **Description** Id — An integer indicating the Keyfactor Command reference ID for the certificate collection. Name — A string containing the name of the certificate collection. For more information about certificate collections, see Saving Search Criteria as a Collection on page 42.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>1</td><td>ExpirationLogger</td></tr>
<tr><td>2</td><td>ExpirationPowershell</td></tr>
<tr><td>3</td><td>ExpirationRenewal</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• LogTarget This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Expiration

The PUT /Alerts/Expiration method is used to update an expiration alert. This method returns HTTP 200 OK on a success with details about the alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 151: PUT Alerts Expiration Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the expiration alert.
DisplayName	Body	**Required**. A string indicating the display name for the expiration alert. This name appears in the Expiration Certificate Request Alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {cn} in the alert definition and each alert generated at processing time will contain the specific common name of the given certificate instead of the variable {cn}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: “Hello {requester:givenname},\n\nThe certificate in the name {cn} issued on {certnotbefore} from {CAreqID} using the {template} template will expire on {certnotafter}. If this certificate is still in use, please consider getting a new one.\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Serial Number: {serial}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>Thumbprint: {thumbprint}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System” See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings.

Name	In	Description
		Note: The $(requester:givenname) substitutable special text token is only supported in environments using Active Directory as an identity provider.
ExpirationWarningDays	Body	**Required**. An integer indicating the number of days prior to expiration to send the warning. **Example:** When alerts run, the alert engine reports on all the certificates expiring within the next X days (e.g. 30 days) from the execution time that have not previously been reported on. This means that if the alerts run daily and have been running daily regularly for some time, only a single day of expiring certificates will be reported on by any given alert run. For example, say you create a new alert that has never run before for collection A and set it to 30 days. You configure it to run daily at 5:00 am. The alert runs for the first time at 5:00 am on July 1st. All the certificates in collection A that will expire between July 1st at 12:oo am UTC and July 31 at 12:00 am UTC will be alerted on. The next day when the alert runs again at 5:00 am on July 2nd, the certificates in collection A expiring between July 31st at 12:00 am UTC and August 1st at 12:00 am UTC will be alerted on. If alerts are missed for a period of time (due to an outage, for example), the next run of the alerts will check the previous successful run date for the alerts and report on certificates expiring X days from that outage window. For example, using the collection A alert referenced above, say an outage caused the alerts not to run on August 1 and August 2. On August 3, the alert would run again at 5:00 am, and the certificates in collection A expiring between August 30th at 12:00 am UTC and September 2nd at 12:00 am UTC would be alerted on.
Recipients	Body	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail} The certificate requester, based on a lookup in Active Directory of the email address associated with the requester

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>on the certificate.

Note: The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider.

• Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQueryId</td><td>Body</td><td>Required. An integer indicating the certificate collection on which to base the alert.
Use the GET /CertificateCollections method (see <a>GET Certificate Collections on page 1296) to retrieve a list of all the certificate collections to determine the collection ID.</td></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.

<table><tr><th>ID</th><th>Event Handler Type</th></tr><tr><td>1</td><td>ExpirationLogger</td></tr><tr><td>2</td><td>ExpirationPowershell</td></tr><tr><td>3</td><td>ExpirationRenewal</td></tr></table></td></tr><tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr></table>
For more information about event handlers, see <a>Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:</td></tr>
</table>

Name	In	Description

Value	Description
Id	An integer indicating the Keyfactor Command reference ID of the configured parameter.
Key	A string indicating the reference name of the configured parameter.
DefaultValue	A string indicating the value for the parameter. This value is related to the type of parameter (see *ParameterType*).
ParameterType	A string containing the parameter type. Supported types are: • LogTarget This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings.

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		   {
		      "Id": 28,
		      "Key": "cn",
		      "DefaultValue": "cn",
		      "ParameterType": "Token"
		   },
		   {
		      "Id": 29,
		      "Key": "AppOwnerFirstName",
		      "DefaultValue": "metadata:AppOwnerFirstName",
		      "ParameterType": "Token"
		   },
		   {
		      "Id": 30,
		      "Key": "Text",
		      "DefaultValue": "Expiration Alert: Enterprise Web Server",
		      "ParameterType": "Value"
		   },
		   {
		      "Id": 32,
		      "Key": "ScriptName",
		      "DefaultValue": "MyScript.ps1",
		      "ParameterType": "Script"
		   }
		]</pre>

Table 152: PUT Alerts Expiration Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the expiration alert.
DisplayName	A string indicating the display name for the expiration alert. This name appears in the Expiration Certificate Request Alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings.
ExpirationWarningDays	An integer indicating the number of days prior to expiration to send the warning. **Example:** When alerts run, the alert engine reports on all the certificates expiring within the next X days (e.g. 30 days) from the execution time that have not previously been reported on. This means that if the alerts run daily and have been running daily regularly for some time, only a single day of expiring certificates will be reported on by any given alert run. For example, say you create a new alert that has never run before for collection A and set it to 30 days. You configure it to run daily at 5:00 am. The alert runs for the first time at 5:00 am on July 1st. All the certificates in collection A that will expire between July 1st at 12:oo am UTC and July 31 at 12:00 am UTC will be alerted on. The next day when the alert runs again at 5:00 am on July 2nd, the certificates in collection A expiring between July 31st at 12:00 am UTC and August 1st at 12:00 am UTC will be alerted on. If alerts are missed for a period of time (due to an outage, for example), the next run of the alerts will check the previous successful run date for the alerts and report on certificates expiring X days from that outage window. For example, using the collection A alert referenced above, say an outage caused the alerts not to run on August 1 and August 2. On August 3, the alert would run again at 5:00 am, and the certificates in collection A expiring between August 30th at 12:00 am UTC and September 2nd at 12:00 am UTC would be alerted on.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Recipients</td><td>An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include:
{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQuery</td><td>An object indicating the certificate collection on which the alert is based. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the certificate collection.</td></tr><tr><td>Name</td><td>A string containing the name of the certificate collection.</td></tr></table>For more information about certificate collections, see Saving Search Criteria as a Collection on page 42.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>1</td><td>ExpirationLogger</td></tr>
<tr><td>2</td><td>ExpirationPowershell</td></tr>
<tr><td>3</td><td>ExpirationRenewal</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• LogTarget This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 4: Substitutable Special Text for Expiration Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Expiration Test

The POST /Alerts/Expiration/Test method is used to test individual certificate expiration alerts. This method returns HTTP 200 OK on a success with details about the resulting alerts generated or a response of “NoActionTaken” if no certificates match the test criteria entered.

Tip: Alerts are generated when a certificate has expired or is approaching expiration as defined by the timeframe configured in the alert.
By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Expiration Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If more than 100 alerts are generated, no email messages will be sent and you'll have the opportunity to view the first 100 alerts generated.

If you're using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message written, certificates renewed) when the test is run. This is true regardless of the setting of the *SendAlerts* flag.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 153: POST Alerts Expiration Test Input Parameters

Name	In	Description
AlertId	Body	n integer indicating the reference ID of expiration alert to test. Use the GET /Alerts/Expiration method (see GET Alerts Expiration on page 936) to retrieve a list of all your expiration alerts to determine the alert Id.
EvaluationDate	Body	A string indicating the start date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). You can use the date range to simulate running the alerts a month from now instead of today, for example, or put in a broad date range to be sure you pick up some expiring certificates for testing purposes.
PreviousEvaluationDate	Body	A string indicating the end date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
SendAlerts	Body	A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is *false*.

Table 154: POST Alerts Expiration Test Response Data

Parameter	Description
ExpirationAlerts	An object containing alert details resulting from the test. Expiration alert details are:
AlertBuildResult	A string indicating the outcome of the test (e.g. Success).

Name	Description
CAName	A string indicating the certificate authority that issued the certificate in hostname\logical name format.
CARow	An integer containing the CA's reference ID for certificate.
IssuedCN	A string indicating the common name of the certificate.
Expiry	A string indicating the date and time when the certificate expires.
Subject	A string indicating the subject for the email message, including any replaced substitutable special text.
Message	A string indicating the email message, including any replaced substitutable special text. See Table 4: Substitutable Special Text for Expiration Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert.
SendDate	A string indicating the date on which the alert will be sent, based on configuration of the *ExpirationWarningDays* in the alert (e.g. if the alert is configured for one month before expiration and the certificate expires on July 20, the alert will have a send date of June 20).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Expiration Test All

The POST /Alerts/Expiration/TestAll method is used to test all certificate expiration alerts. This method returns HTTP 200 OK on a success with details about the resulting alerts generated or a response of “NoActionTaken” if no certificates match the test criteria entered.

Tip: Alerts are generated when a certificate has expired or is approaching expiration as defined by the timeframe configured in the alert.
By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Expiration Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If more than 100 alerts are generated, no email messages will be sent and you’ll have the opportunity to view the first 100 alerts generated.

If you’re using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message written, certificates renewed) when the test is run. This is true regardless of the setting of the *SendAlerts* flag.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 155: POST Alerts Expiration Test All Input Parameters

Name	In	Description
EvaluationDate	Body	A string indicating the start date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). You can use the date range to simulate running the alerts a month from now instead of today, for example, or put in a broad date range to be sure you pick up some expiring certificates for testing purposes.
PreviousEvaluationDate	Body	A string indicating the end date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
SendAlerts	Body	A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is *false*.

Table 156: POST Alerts Expiration Test All Response Data

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>ExpirationAlerts</td><td>An object containing alert details resulting from the test. Expiration alert details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CAName</td><td>A string indicating the certificate authority that issued the certificate in hostname\logical name format.</td></tr>
<tr><td>CARow</td><td>An integer containing the CA's reference ID for certificate.</td></tr>
<tr><td>IssuedCN</td><td>A string indicating the common name of the certificate.</td></tr>
<tr><td>Expiry</td><td>A string indicating the date and time when the certificate expires.</td></tr>
<tr><td>Subject</td><td>A string indicating the subject for the email message, including any replaced substitutable special text.</td></tr>
<tr><td>Message</td><td>A string indicating the email message, including any replaced substitutable special text.
See Table 4: Substitutable Special Text for Expiration Alerts in the Keyfactor Command Reference Guide for a complete list of available substitutable special text strings.</td></tr>
<tr><td>Recipients</td><td>An array of strings containing the recipients for the alert.</td></tr>
<tr><td>SendDate</td><td>A string indicating the date on which the alert will be sent, based on configuration of the ExpirationWarningDays in the alert (e.g. if the alert is configured for one month before expiration and the certificate expires on July 20, the alert will have a send date of June 20).</td></tr>
</table></td></tr>
<tr><td>AlertBuildResult</td><td>A string indicating the outcome of the test (e.g. Success).</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.4.3 Alerts Issued

The Alerts Issued component of the Keyfactor API includes methods necessary to create, update, retrieve, schedule, and delete alerts for issued certificate requests.

Table 157: Alerts Issued

Endpoint	Method	Description	Link
/Alerts/Issued/{id}	DELETE	Deletes an issued certificate request alert for the specified ID.	DELETE Alerts Issued ID below
/Alerts/Issued/{id}	GET	Retrieves details for an issued certificate request alert for the specified ID.	GET Alerts Issued ID on the next page
/Alerts/Issued/Schedule	GET	Retrieves details of the schedule for delivery of issued certificate request alerts.	GET Alerts Issued Schedule on page 971
/Alerts/Issued/Schedule	PUT	Updates the schedule for delivery of issued certificate request alerts.	PUT Alerts Issued Schedule on page 973
/Alerts/Issued	GET	Retrieves details for all configured issued certificate request alerts.	GET Alerts Issued on page 976
/Alerts/Issued	POST	Creates a new issued certificate request alert.	POST Alerts Issued on page 981
/Alerts/Issued	PUT	Updates an issued certificate request alert for the specified ID.	PUT Alerts Issued on page 991

DELETE Alerts Issued ID

The DELETE /Alerts/Issued/{id} method is used to delete the issued certificate request alert with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 158: DELETE Alerts Issued {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the issued certificate request alert to be deleted. Use the *GET /Alerts/Issued* method (see GET Alerts Issued on page 976) to retrieve a list of all the issued request alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Issued ID

The GET /Alerts/Issued/{id} method is used to retrieve details for the issued certificate request alerts with the specified ID. This method returns HTTP 200 OK on a success with details about the specified issued certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 159: GET Alerts Issued {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the issued certificate request alert. Use the *GET /Alerts/Issued* method (see GET Alerts Issued on page 976) to retrieve a list of all the issued request alerts to determine the alert ID.

Table 160: GET Alerts Issued {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the issued request alert.
DisplayName	A string indicating the display name for the issued request alert. This name appears in the issued request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nThe certificate you requested in the name {cn} was successfully issued on {certnotbefore}. You can download it from here:\n\n{dnldlink}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Serial Number: {serial}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>Thumbprint: {thumbprint}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 6: Substitutable Special Text for Issued Certificate Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>text strings include:
{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the issued request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all issued certificate requests. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for All Templates.</td></tr><tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr><tr><td>ForestRoot</td><td>A string indicating the forest root of the template.

Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr><tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>4</td><td>IssuedLogger</td></tr>
<tr><td>5</td><td>IssuedPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggeredIt is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 6: Substitutable Special Text for Issued Certificate Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Issued Schedule

The GET /Alerts/Issued/Schedule method is used to retrieve the schedule for delivery of issued certificate request alerts configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the schedule for issued certificate request alerts. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 161: GET Alerts Issued Schedule Response Data

Name	Description
Schedule	An object indicating the schedule for delivery of the issued request alerts. Possible values are:

Name	Description
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Issued Schedule

The PUT /Alerts/Issued/Schedule method is used to create or update the schedule for delivery of issued certificate request alerts configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the schedule for issued certificate request alerts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 162: PUT Alerts Issued Schedule Input Parameters

Name	In	Description

Schedule	Body	An object indicating the schedule for delivery of the issued request alerts. Possible values are:
		
		Interval
		A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
		
		Minutes
		An integer indicating the number of minutes between each interval.
		
		For example, every hour:
		
		`"Interval": { "Minutes": 60 }`
		
		Daily
		A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
		
		Time
		The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
		
		For example, daily at 11:30 pm:
		
		`"Daily": { "Time": "2023-11-25T23:30:00Z" }`

Table 163: PUT Alerts Issued Schedule Response Data

Name	Description
Schedule	An object indicating the schedule for delivery of the issued request alerts. Possible values are:

Name	Description
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Issued

The GET /Alerts/Issued method is used to retrieve details of all issued certificate request alerts configured in Keyfactor Command. Results can be limited to selected alerts using filtering, and URL parameters can be used to specify paging and sorting. This method returns HTTP 200 OK on a success with details about the specified issued certificate request alerts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 164: GET Alerts Issued Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • DisplayName • Message • RegisteredEventHandlerId • Subject • Template_Id • UseHandler
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 165: GET Alerts Issued Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the issued request alert.
DisplayName	A string indicating the display name for the issued request alert. This name appears in the issued request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nThe certificate you requested in the name {cn} was successfully issued on {certnotbefore}. You can download it from here:\n\n{dnldlink}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Serial Number: {serial}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>Thumbprint: {thumbprint}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 6: Substitutable Special Text for Issued Certificate Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>text strings include:
{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the issued request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all issued certificate requests. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for All Templates.</td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr>
<tr><td>ForestRoot</td><td>A string indicating the forest root of the template.

Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr>
<tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>4</td><td>IssuedLogger</td></tr>
<tr><td>5</td><td>IssuedPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggeredIt is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 6: Substitutable Special Text for Issued Certificate Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Issued

The POST /Alerts/Issued method is used to create a new issued certificate request alert. This method returns HTTP 200 OK on a success with details about the issued certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 166: POST Alerts Issued Input Parameters

Name	In	Description
DisplayName	Body	**Required**. A string indicating the display name for the issued request alert. This name appears in the issued request alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nThe certificate you requested in the name {cn} was successfully issued on {certnotbefore}. You can download it from here:\n\n{dnldlink}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Serial Number: {serial}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>Thumbprint: {thumbprint}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 6: Substitutable Special Text for Issued Certificate Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings. **Note:** The $(requester:givenname) substitutable special text token is only supported in environments using Active

Name	In	Description
		Directory as an identity provider.
Recipients	Body	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail} The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.
TemplateId	Body	An integer indicating the certificate template for which the issued request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all issued certificate requests. Use the GET /Templates method (see GET Templates on page 2422) to retrieve a list of all the templates to determine the template ID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.<table><tr><th>ID</th><th>Event Handler Type</th></tr><tr><td>4</td><td>IssuedLogger</td></tr><tr><td>5</td><td>IssuedPowershell</td></tr></table></td></tr><tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr></table>For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr><tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr><tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr><tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:</td></tr></table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• LogTarget
		This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged.
		• Script
		This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggeredIt is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
		• Token
		This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 6: Substitutable Special Text for Issued Certificate Alerts for a complete list of available substitutable special text strings.
		• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		  {</pre>

Name	In	Description

		"Id": 28,
		"Key": "cn",
		"DefaultValue": "cn",
		"ParameterType": "Token"
		},
		{
		"Id": 29,
		"Key": "AppOwnerFirstName",
		"DefaultValue": "metadata:AppOwnerFirstName",
		"ParameterType": "Token"
		},
		{
		"Id": 30,
		"Key": "Text",
		"DefaultValue": "Issued Alert: Enterprise Web Server",
		"ParameterType": "Value"
		},
		{
		"Id": 31,
		"Key": "DownloadLink",
		"DefaultValue": "dnldlink",
		"ParameterType": "Token"
		},
		{
		"Id": 32,
		"Key": "ScriptName",
		"DefaultValue": "MyScript.ps1",
		"ParameterType": "Script"
		}
		]

Table 167: POST Alerts Issued Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the issued request alert.
DisplayName	A string indicating the display name for the issued request alert. This name appears in the issued request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nThe certificate you requested in the name {cn} was successfully issued on {certnotbefore}. You can download it from here:\n\n{dnldlink}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Serial Number: {serial}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>Thumbprint: {thumbprint}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 6: Substitutable Special Text for Issued Certificate Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>text strings include:
{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the issued request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all issued certificate requests. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for All Templates.</td></tr><tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr><tr><td>ForestRoot</td><td>A string indicating the forest root of the template.
Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr><tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>4</td><td>IssuedLogger</td></tr>
<tr><td>5</td><td>IssuedPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged.
• Script
This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggeredIt is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
• Token
This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 6: Substitutable Special Text for Issued Certificate Alerts for a complete list of available substitutable special text strings.
• Value
This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr>
</table>
</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Issued

The PUT /Alerts/Issued method is used to update an issued certificate request alert. This method returns HTTP 200 OK on a success with details about the issued certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 168: PUT Alerts Issued Input Parameters

Name	In	Description
id	Path	An integer indicating the Keyfactor Command reference ID of the issued request alert.
DisplayName	Body	**Required**. A string indicating the display name for the issued request alert. This name appears in the issued request alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nThe certificate you requested in the name {cn} was successfully issued on {certnotbefore}. You can download it from here:\n\n{dnldlink}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Serial Number: {serial}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>Thumbprint: {thumbprint}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 6: Substitutable Special Text for Issued Certificate Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings.

Name	In	Description
		Note: The $(requester:givenname) substitutable special text token is only supported in environments using Active Directory as an identity provider.
Recipients	Body	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail} The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.
TemplateId	Body	An integer indicating the certificate template for which the issued request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all issued certificate requests. Use the GET /Templates method (see GET Templates on page 2422) to retrieve a list of all the templates to determine the template ID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>4</td><td>IssuedLogger</td></tr>
<tr><td>5</td><td>IssuedPowershell</td></tr>
</table></td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:</td></tr>
</table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>• LogTarget
		This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged.
		• Script
		This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggeredIt is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
		• Token
		This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 6: Substitutable Special Text for Issued Certificate Alerts for a complete list of available substitutable special text strings.
		• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		  {</pre>

Name	In	Description

		"Id": 28,
		"Key": "cn",
		"DefaultValue": "cn",
		"ParameterType": "Token"
		},
		{
		"Id": 29,
		"Key": "AppOwnerFirstName",
		"DefaultValue": "metadata:AppOwnerFirstName",
		"ParameterType": "Token"
		},
		{
		"Id": 30,
		"Key": "Text",
		"DefaultValue": "Issued Alert: Enterprise Web Server",
		"ParameterType": "Value"
		},
		{
		"Id": 31,
		"Key": "DownloadLink",
		"DefaultValue": "dnldlink",
		"ParameterType": "Token"
		},
		{
		"Id": 32,
		"Key": "ScriptName",
		"DefaultValue": "MyScript.ps1",
		"ParameterType": "Script"
		}
		]

Table 169: PUT Alerts Issued Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the issued request alert.
DisplayName	A string indicating the display name for the issued request alert. This name appears in the issued request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello {requester:givenname},\n\nThe certificate you requested in the name {cn} was successfully issued on {certnotbefore}. You can download it from here:\n\n{dnldlink}\n\nCertificate information includes:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>Serial Number: {serial}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>Thumbprint: {thumbprint}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td>DN: {dn}</td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nThanks!\n\nYour Certificate Management System" See Table 6: Substitutable Special Text for Issued Certificate Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>text strings include:
{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the issued request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all issued certificate requests. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for All Templates.</td></tr><tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr><tr><td>ForestRoot</td><td>A string indicating the forest root of the template.

Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr><tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>4</td><td>IssuedLogger</td></tr>
<tr><td>5</td><td>IssuedPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggeredIt is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 6: Substitutable Special Text for Issued Certificate Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.4.4 Alerts Key Rotation

The Alerts Key Rotation component of the Keyfactor API includes methods necessary to create, update, retrieve, schedule, and delete alerts for SSH keys approaching the end of the key lifetime. The default key lifetime is 365 days, but this setting is configurable (see Application Settings: SSH Tab on page 620). Key rotation alerts apply to both user keys (see My SSH Key on page 531) and

service account keys (see Service Account Keys on page 542) generated within Keyfactor Command.

Table 170: Alerts Key Rotation

Endpoint	Method	Description	Link
/Alerts/KeyRotation/{id}	DELETE	Deletes an SSH key rotation alert for the specified ID.	DELETE Alerts Key Rotation ID below
/Alerts/KeyRotation/{id}	GET	Retrieves details for the SSH key rotation alert for the specified ID.	GET Alerts Key Rotation ID on the next page
/Alerts/KeyRotation/Schedule	GET	Retrieves details of the schedule for delivery of SSH key rotation alerts.	GET Alerts Key Rotation Schedule on page 1006
/Alerts/KeyRotation/Schedule	PUT	Updates the schedule for delivery of SSH key rotation alerts.	PUT Alerts Key Rotation Schedule on page 1008
/Alerts/KeyRotation	GET	Retrieves details for all configured SSH key rotation alerts.	GET Alerts Key Rotation on page 1011
/Alerts/KeyRotation	POST	Creates a new SSH key rotation alert.	POST Alerts Key Rotation on page 1015
/Alerts/KeyRotation	PUT	Updates the SSH key rotation alert for a specified ID.	PUT Alerts Key Rotation on page 1022
/Alerts/KeyRotation/Test	POST	Used to test specific SSH key rotation alerts.	POST Alerts Key Rotation Test on page 1030
/Alerts/KeyRotation/TestAll	POST	Used to test all SSH key rotation alerts.	POST Alerts Key Rotation Test All on page 1032

DELETE Alerts Key Rotation ID

The DELTE /Alerts/KeyRotation/{id} method is used to delete the SSH key rotation alert with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 171: DELETE Alerts Key Rotation {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH key rotation alert to be deleted. Use the GET /Alerts/KeyRotation method (see GET Alerts Key Rotation on page 1011) to retrieve a list of all the SSH key rotation alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Key Rotation ID

The GET /Alerts/KeyRotation/{id} method is used to retrieve details for the SSH key rotation alerts with the specified ID. This method returns HTTP 200 OK on a success with details about the specified SSH key rotation alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 172: GET Alerts Key Rotation {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH key rotation alert. Use the GET /Alerts/KeyRotation method (see GET Alerts Key Rotation on page 1011) to retrieve a list of all the SSH key rotation alerts to determine the alert ID.

Table 173: GET Alerts Key Rotation {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH key rotation alert.
DisplayName	A string indicating the display name for the SSH key rotation alert. This name appears in the SSH key rotation alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the key record at processing time. For example, you can enter {comment} in the alert definition and each alert generated at processing time will contain the specific key comment of the given SSH key instead of the variable {comment}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nYou requested an SSH key pair almost a year ago with the following information:\n\n<table>\n<tr><th>Field</th><th>Value</th></tr>\n<tr><td>Username</td><td>{username}</td></tr>\n<tr><td>Fingerprint</td><td>{fingerprint}</td></tr>\n<tr><td>Comment</td><td>{comment}</td></tr>\n<tr><td>Key Length</td><td>{keylength}</td></tr>\n<tr><td>Key Type</td><td>{keytype}</td></tr>\n<tr><td>Number of Server Logons for Key</td><td>{serverlogons}</td></tr>\n</table>\n\nCorporate policy requires key rotation every year. Please visit the My SSH Key Portal for user keys or the Service Account Key Portal for service account keys and request a new key pair.\n\nThanks!" See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings.
RotationWarningDays	An integer indicating the number of days prior to the end of an SSH key's lifetime the alert should be triggered.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>10</td><td>SSHKeyRotationLogger</td></tr>
<tr><td>11</td><td>SSHKeyRotationPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Key Rotation Schedule

The GET /Alerts/KeyRotation/Schedule method is used to retrieve the schedule for delivery of SSH key rotation alerts configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the schedule for SSH key rotation alerts. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 174: GET Alerts Key Rotation Schedule Response Data

Name	Description

Schedule	An object indicating the schedule for delivery of the SSH key rotation alerts. Possible values are:
	
	Interval
	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
	
	Minutes
	An integer indicating the number of minutes between each interval.
	
	For example, every hour:
	
	`"Interval": { "Minutes": 60 }`
	
	Daily
	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
	
	Time
	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
	
	For example, daily at 11:30 pm:
	
	`"Daily": { "Time": "2023-11-25T23:30:00Z" }`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Key Rotation Schedule

The PUT /Alerts/KeyRotation/Schedule method is used to create or update the schedule for delivery of SSH key rotation alerts configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the schedule for SSH key rotation alerts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 175: PUT Alerts Key Rotation Schedule Input Parameters

Name	In	Description

Schedule	Body	An object indicating the schedule for delivery of the SSH key rotation alerts. Possible values are:
		
		Interval
		A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
		
		Minutes
		An integer indicating the number of minutes between each interval.
		
		For example, every hour:
		
		`"Interval": { "Minutes": 60 }`
		
		Daily
		A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
		
		Time
		The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
		
		For example, daily at 11:30 pm:
		
		`"Daily": { "Time": "2023-11-25T23:30:00Z" }`

Table 176: PUT Alerts Key Rotation Schedule Response Data

Name	Description
Schedule	An object indicating the schedule for delivery of the SSH key rotation alerts. Possible values are:

Name	Description
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Key Rotation

The GET /Alerts/KeyRotation method is used to retrieve details of all SSH key rotation alerts configured in Keyfactor Command. Results can be limited to selected alerts using filtering, and URL parameters can be used to specify paging and sorting. This method returns HTTP 200 OK on a success with details about the specified SSH key rotation alerts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 177: GET Alerts Key Rotation Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • Days • DisplayName • Message • RegisteredEventHandlerId • ScheduledTaskId • Subject • UseHandler
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 178: GET Alerts Key Rotation Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH key rotation alert.
DisplayName	A string indicating the display name for the SSH key rotation alert. This name appears in the SSH key rotation alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the key record at processing time. For example, you can enter {comment} in the alert definition and each alert generated at processing time will contain the specific key comment of the given SSH key instead of the variable {comment}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nYou requested an SSH key pair almost a year ago with the following information:\n\n<table>\n<tr><th>Field</th><th>Value</th></tr>\n<tr><td>Username</td><td>{username}</td></tr>\n<tr><td>Fingerprint</td><td>{fingerprint}</td></tr>\n<tr><td>Comment</td><td>{comment}</td></tr>\n<tr><td>Key Length</td><td>{keylength}</td></tr>\n<tr><td>Key Type</td><td>{keytype}</td></tr>\n<tr><td>Number of Server Logons for Key</td><td>{serverlogons}</td></tr>\n</table>\n\nCorporate policy requires key rotation every year. Please visit the My SSH Key Portal for user keys or the Service Account Key Portal for service account keys and request a new key pair.\n\nThanks!" See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings.
RotationWarningDays	An integer indicating the number of days prior to the end of an SSH key's lifetime the alert should be triggered.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>10</td><td>SSHKeyRotationLogger</td></tr>
<tr><td>11</td><td>SSHKeyRotationPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Key Rotation

The POST /Alerts/KeyRotation method is used to create a new SSH key rotation alert. This method returns HTTP 200 OK on a success with details about the SSH key rotation alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 179: POST Alerts Key Rotation Input Parameters

Name	In	Description
DisplayName	Body	**Required**. A string indicating the display name for the SSH key rotation alert. This name appears in the SSH key rotation alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the key record at processing time. For example, you can enter {comment} in the alert definition and each alert generated at processing time will contain the specific key comment of the given SSH key instead of the variable {comment}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nYou requested an SSH key pair almost a year ago with the following inform- ation:\n\n<t- able>\n<tr><th>Field</th><th>Value</th></tr>\n<tr><t- d>Username</td><td> {username}</td></tr>\n<tr><td>Fingerprint</td><td> {fingerprint}</td></tr>\n<tr><td>Comment</td><td>{comment}</t- d></tr>\n<tr><td>Key Length</td><td>{keylength}</t- d></tr>\n<tr><td>Key Type</td><td> {keytype}</td></tr>\n<tr><td>Number of Server Logons for Key</t- d><td>{serverlogons}</td></tr>\n</table>\n\nCorporate policy requires key rotation every year. Please visit the My SSH Key Portal for user keys or the Service Account Key Portal for service account keys and request a new key pair.\n\nThanks!" See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings.
Rota- tionWarningDays	Body	An integer indicating the number of days prior to the end of an SSH key's lifetime the alert should be triggered.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table><tr><th>ID</th><th>Event Handler Type</th></tr><tr><td>10</td><td>SSHKeyRotationLogger</td></tr><tr><td>11</td><td>SSHKeyRotationPowershell</td></tr></table></td></tr><tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr></table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr><tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr><tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr><tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
• LogTarget
This type is used for the event logging handler and is used to reference the fully</td></tr></table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>qualified domain name of the target machine to which event should be logged.
		• Script
		This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
		• Token
		This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings.
		• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		  {
		    "Id": 28,
		    "Key": "user",
		    "DefaultValue": "username",
		    "ParameterType": "Token"
		  },
		  {
		    "Id": 29,
		    "Key": "comment",
		    "DefaultValue": "comment",
		    "ParameterType": "Token"
		  },</pre>

Name	In	Description

		[
		{
		"Id": 30,
		"Key": "Text",
		"DefaultValue": "Key Rotation Alert: 3 Days",
		"ParameterType": "Value"
		},
		{
		"Id": 32,
		"Key": "ScriptName",
		"DefaultValue": "MyScript.ps1",
		"ParameterType": "Script"
		}
		]

Table 180: POST Alerts Key Rotation Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH key rotation alert.
DisplayName	A string indicating the display name for the SSH key rotation alert. This name appears in the SSH key rotation alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the key record at processing time. For example, you can enter {comment} in the alert definition and each alert generated at processing time will contain the specific key comment of the given SSH key instead of the variable {comment}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nYou requested an SSH key pair almost a year ago with the following information:\n\n<table>\n<tr><th>Field</th><th>Value</th></tr>\n<tr><td>Username</td><td>{username}</td></tr>\n<tr><td>Fingerprint</td><td>{fingerprint}</td></tr>\n<tr><td>Comment</td><td>{comment}</td></tr>\n<tr><td>Key Length</td><td>{keylength}</td></tr>\n<tr><td>Key Type</td><td>{keytype}</td></tr>\n<tr><td>Number of Server Logons for Key</td><td>{serverlogons}</td></tr>\n</table>\n\nCorporate policy requires key rotation every year. Please visit the My SSH Key Portal for user keys or the Service Account Key Portal for service account keys and request a new key pair.\n\nThanks!" See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings.
RotationWarningDays	An integer indicating the number of days prior to the end of an SSH key's lifetime the alert should be triggered.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>10</td><td>SSHKeyRotationLogger</td></tr>
<tr><td>11</td><td>SSHKeyRotationPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Key Rotation

The PUT /Alerts/KeyRotation method is used to update a SSH key rotation alert. This method returns HTTP 200 OK on a success with details about the SSH key rotation alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 181: PUT Alerts Key Rotation Input Parameters

Name	In	Description
id	Path	An integer indicating the Keyfactor Command reference ID of the SSH key rotation alert.
DisplayName	Body	**Required**. A string indicating the display name for the SSH key rotation alert. This name appears in the SSH key rotation alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the key record at processing time. For example, you can enter {comment} in the alert definition and each alert generated at processing time will contain the specific key comment of the given SSH key instead of the variable {comment}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nYou requested an SSH key pair almost a year ago with the following inform- ation:\n\n<t- able>\n<tr><th>Field</th><th>Value</th></tr>\n<tr><t- d>Username</td><td> {username}</td></tr>\n<tr><td>Fingerprint</td><td> {fingerprint}</td></tr>\n<tr><td>Comment</td><td>{comment}</t- d></tr>\n<tr><td>Key Length</td><td>{keylength}</t- d></tr>\n<tr><td>Key Type</td><td> {keytype}</td></tr>\n<tr><td>Number of Server Logons for Key</t- d><td>{serverlogons}</td></tr>\n</table>\n\nCorporate policy requires key rotation every year. Please visit the My SSH Key Portal for user keys or the Service Account Key Portal for service account keys and request a new key pair.\n\nThanks!" See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RotationWarningDays</td><td>Body</td><td>An integer indicating the number of days prior to the end of an SSH key's lifetime the alert should be triggered.</td></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table><tr><th>ID</th><th>Event Handler Type</th></tr><tr><td>10</td><td>SSHKeyRotationLogger</td></tr><tr><td>11</td><td>SSHKeyRotationPowershell</td></tr></table></td></tr><tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr><tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr></table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr><tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr><tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr></table></td></tr>
</table>

Name	In	Description

Value	Description
ParameterType	A string containing the parameter type. Supported types are: • LogTarget This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.

For example, for a PowerShell handler:

```
"EventHandlerParameters": [
   {
      "Id": 28,
      "Key": "user",
      "DefaultValue": "username",
      "ParameterType": "Token"
   },
```

Name	In	Description

		[
		{
		"Id": 29,
		"Key": "comment",
		"DefaultValue": "comment",
		"ParameterType": "Token"
		},
		{
		"Id": 30,
		"Key": "Text",
		"DefaultValue": "Key Rotation Alert: 3 Days",
		"ParameterType": "Value"
		},
		{
		"Id": 32,
		"Key": "ScriptName",
		"DefaultValue": "MyScript.ps1",
		"ParameterType": "Script"
		}
		]

Table 182: PUT Alerts Key Rotation Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH key rotation alert.
DisplayName	A string indicating the display name for the SSH key rotation alert. This name appears in the SSH key rotation alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the key record at processing time. For example, you can enter {comment} in the alert definition and each alert generated at processing time will contain the specific key comment of the given SSH key instead of the variable {comment}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nYou requested an SSH key pair almost a year ago with the following information:\n\n<table>\n<tr><th>Field</th><th>Value</th></tr>\n<tr><td>Username</td><td>{username}</td></tr>\n<tr><td>Fingerprint</td><td>{fingerprint}</td></tr>\n<tr><td>Comment</td><td>{comment}</td></tr>\n<tr><td>Key Length</td><td>{keylength}</td></tr>\n<tr><td>Key Type</td><td>{keytype}</td></tr>\n<tr><td>Number of Server Logons for Key</td><td>{serverlogons}</td></tr>\n</table>\n\nCorporate policy requires key rotation every year. Please visit the My SSH Key Portal for user keys or the Service Account Key Portal for service account keys and request a new key pair.\n\nThanks!" See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings.
RotationWarningDays	An integer indicating the number of days prior to the end of an SSH key's lifetime the alert should be triggered.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>10</td><td>SSHKeyRotationLogger</td></tr>
<tr><td>11</td><td>SSHKeyRotationPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 8: Substitutable Special Text for Key Rotation Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Key Rotation Test

The POST /Alerts/KeyRotation/Test method is used to test a specific SSH key rotation alert. This method returns HTTP 200 OK on a success with details about the SSH key rotation alert or a response of “NoActionTaken” if no keys match the test criteria entered.

Tip: Alerts are generated when an SSH key is approaching or has reached its stale date as defined by the timeframe configured in the alert and the SSH key lifetime (the *Key Lifetime (days)* application setting).

By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Key Rotation Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If more than 100 alerts are generated, no email messages will be sent and you'll have the opportunity to view the first 100 alerts generated.

If you're using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message written) when the test is run. This is true regardless of the setting of the *SendAlerts* flag.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 183: POST Alerts Key Rotation Test Input Parameters

Parameter	In	Description
AlertId	Body	**Required**. An integer of the reference ID of the SSH key rotation alert to test. Use the GET /Alerts/KeyRotation method (see GET Alerts Key Rotation on page 1011) to retrieve a list of all your key rotation alerts to determine the alert Id.
EvaluationDate	Body	**Required**. A string indicating the start date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). You can use the date range to simulate running the alerts a month from now instead of today, for example, or put in a broad date range to be sure you pick up some expiring keys for testing purposes.
PreviousEvaluationDate	Body	**Required**. A string indicating the end date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
SendAlerts	Body	A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is *false*.

Table 184: POST Alerts Key Rotation Test Response Data

Parameter	Description
KeyRotationAlerts	An object containing alert details resulting from the test. Expiration alert details are: **Name** / **Description** Subject — A string indicating the subject for the email message, including any replaced substitutable special text. Message — A string indicating the email message, including any replaced substitutable special text See Table 8: Substitutable Special Text for Key Rotation Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings. Recipient — A string indicating the recipient for the alert.
AlertBuildResult	A string indicating the outcome of the test (e.g. Success).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Key Rotation Test All

The POST /Alerts/KeyRotation/TestAll method is used to test all SSH key rotation alerts. This method returns HTTP 200 OK on a success with details about the SSH key rotation alert or a response of "NoActionTaken" if no keys match the test criteria entered.

Tip: Alerts are generated when an SSH key is approaching or has reached its stale date as defined by the timeframe configured in the alert and the SSH key lifetime (the *Key Lifetime (days)* application setting).
By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Key Rotation Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If more than 100 alerts are generated, no email messages will be sent and you'll have the opportunity to view the first 100 alerts generated.

If you're using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message written) when the test is run. This is true regardless of the setting of the *SendAlerts* flag.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 185: POST Alerts Key Rotation Test All Input Parameters

Parameter	In	Description
EvaluationDate	Body	**Required**. A string indicating the start date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). You can use the date range to simulate running the alerts a month from now instead of today, for example, or put in a broad date range to be sure you pick up some expiring keys for testing purposes.
PreviousEvaluationDate	Body	**Required**. A string indicating the end date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
SendAlerts	Body	A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is *false*.

Table 186: POST Alerts Key Rotation Test All Response Data

Parameter	Description
KeyRotationAlerts	An object containing alert details resulting from the test. Expiration alert details are: **Name** / **Description** Subject — A string indicating the subject for the email message, including any replaced substitutable special text. Message — A string indicating the email message, including any replaced substitutable special text See Table 8: Substitutable Special Text for Key Rotation Alerts in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings. Recipient — A string indicating the recipient for the alert.
AlertBuildResult	A string indicating the outcome of the test (e.g. Success).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.4.5 Alerts Pending

The Alerts Pending component of the Keyfactor API includes methods necessary to create, update, retrieve, schedule, and delete alerts for certificate requests that require approval based on policy on the CA.

Important: Pending alerts are **not** used to provide email alerts for certificate requests that require approval based on policies configured in Keyfactor Command workflows. These alerts are configured as steps within the workflow (see Workflow Definitions on page 2487). For more information about the difference between alerting for certificate requests that require manager approval at the CA level and alerting for certificate requests that require manager approval at the Keyfactor Command workflow level, see Pending Certificate Request Alerts on page 178.

Table 187: Alerts Pending

Endpoint	Method	Description	Link
/Alerts/Pending/{id}	DELETE	Deletes a pending certificate request alert for the specified ID.	DELETE Alerts Pending ID below
/Alerts/Pending/{id}	GET	Retrieves details for a pending certificate request alert for the specified ID.	GET Alerts Pending ID on the next page
/Alerts/Pending	PUT	Updates a pending certificate request alert for a specified ID.	PUT Alerts Pending on page 1060
/Alerts/Pending/Schedule	GET	Retrieves details of the schedule for delivery of pending certificate request alerts.	GET Alerts Pending Schedule on page 1040
/Alerts/Pending/Schedule	PUT	Updates the schedule for delivery of pending certificate request alerts.	PUT Alerts Pending Schedule on page 1042
/Alerts/Pending	GET	Retrieves details for all configured pending certificate request alerts.	GET Alerts Pending on page 1045
/Alerts/Pending	POST	Creates a new pending certificate request alert.	POST Alerts Pending on page 1050
/Alerts/Pending/Test	POST	Tests all alerts	POST Alerts Pending TestAll on page 1073
/Alerts/Pending/Test/{id}	POST	Tests specific alerts	POST Alerts Pending Test on page 1071

DELETE Alerts Pending ID

The DELTE /Alerts/Pending/{id} method is used to delete the pending certificate request alert with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 188: DELETE Alerts Pending {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the pending certificate request alert to be deleted. Use the GET /Alerts/Pending method (see GET Alerts Pending on page 1045) to retrieve a list of all the pending request alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Pending ID

The GET /Alerts/Pending/{id} method is used to retrieve details for the pending certificate request alerts with the specified ID. This method returns HTTP 200 OK on a success with details about the specified pending certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 189: GET Alerts Pending {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the pending certificate request alert. Use the GET /Alerts/Pending method (see GET Alerts Pending on page 1045) to retrieve a list of all the pending request alerts to determine the alert ID.

Table 190: GET Alerts Pending {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the pending request alert.
DisplayName	A string indicating the display name for the pending request alert. This name appears in the pending request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate request or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nA certificate using the {template} template was requested by {requester:displayname} from {careqid} on {subdate}. The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>DN: {rdn}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td> </td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n {apprlink}\n\nThanks!\n\nYour Certificate Management Tool\n" See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail}

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
• Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the pending request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all pending certificate requests. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for all templates.</td></tr><tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr><tr><td>ForestRoot</td><td>A string indicating the forest root of the template.
Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr><tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>8</td><td>PendingLogger</td></tr>
<tr><td>9</td><td>PendingPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Pending Schedule

The GET /Alerts/Pending/Schedule method is used to retrieve the schedule for delivery of pending certificate request alerts configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the schedule for pending certificate request alerts. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 191: GET Alerts Pending Schedule Response Data

Name	Description

Schedule	An object indicating the schedule for delivery of the pending request alerts. Possible values are:
	
	Interval
	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
	
	Minutes — An integer indicating the number of minutes between each interval.
	
	For example, every hour:
	
	`"Interval": { "Minutes": 60 }`
	
	Daily
	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
	
	Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
	
	For example, daily at 11:30 pm:
	
	`"Daily": { "Time": "2023-11-25T23:30:00Z" }`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Pending Schedule

The PUT /Alerts/Pending/Schedule method is used to create or update the schedule for delivery of pending certificate request alerts configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the schedule for pending certificate request alerts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 192: PUT Alerts Pending Schedule Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Schedule</td><td>Body</td><td>An object indicating the schedule for delivery of the pending request alerts. Possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

Table 193: PUT Alerts Pending Schedule Response Data

Name	Description
Schedule	An object indicating the schedule for delivery of the pending request alerts. Possible values are:

Name	Description
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET Alerts Pending

The GET /Alerts/Pending method is used to retrieve details of all pending certificate request alerts configured in Keyfactor Command. Results can be limited to selected alerts using filtering, and URL parameters can be used to specify paging and sorting. This method returns HTTP 200 OK on a success with details about the specified pending certificate request alerts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 194: GET Alerts Pending Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • DisplayName • Message • RegisteredEventHandlerId • ScheduledTaskId • Subject • Template_Id • UseHandler
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 195: GET Alerts Pending Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the pending request alert.
DisplayName	A string indicating the display name for the pending request alert. This name appears in the pending request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate request or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nA certificate using the {template} template was requested by {requester:displayname} from {careqid} on {subdate}. The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>DN: {rdn}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td> </td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n {apprlink}\n\nThanks!\n\nYour Certificate Management Tool\n" See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail}

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
• Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the pending request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all pending certificate requests. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for all templates.</td></tr><tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr><tr><td>ForestRoot</td><td>A string indicating the forest root of the template.
Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr><tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>8</td><td>PendingLogger</td></tr>
<tr><td>9</td><td>PendingPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Pending

The POST /Alerts/Pending method is used to create a new pending certificate request alert. This method returns HTTP 200 OK on a success with details about the pending certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/monitoring/alerts/modify/

Table 196: POST Alerts Pending Input Parameters

Name	In	Description
DisplayName	Body	**Required**. A string indicating the display name for the pending request alert. This name appears in the pending request alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate request or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nA certificate using the {template} template was requested by {requester:displayname} from {careqid} on {subdate}. The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:ApOwnerFirstName}</td></tr>\n<tr><td>DN: {rdn}</td><td>App Owner Last Name: {metadata:ApOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:ApOwnerEmailAddress}</td></tr>\n<tr><td> </td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n{apprlink}\n\nThanks!\n\nYour Certificate Management Tool\n" See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings. **Note:** The $(requester:givenname) substitutable special text token is only supported in environments using Active Directory as an identity provider.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Recipients</td><td>Body</td><td>An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include:
{requester:mail}
The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.

Note: The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider.
Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>TemplateId</td><td>Body</td><td>An integer indicating the certificate template for which the pending request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all pending certificate requests.
Use the GET /Templates method (see <a>GET Templates on page 2422) to retrieve a list of all the templates to determine the template ID.</td></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.

<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>8</td><td>PendingLogger</td></tr>
<tr><td>9</td><td>PendingPowershell</td></tr>
</table>
</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr><tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr><tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr><tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler and is used to reference the fully qualified domain name of the target machine to which event should be logged.
Script
This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
Token
This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell</td></tr></table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>script. See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings.
		• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		  {
		  "Id": 28,
		  "Key": "cn",
		  "DefaultValue": "rcn",
		  "ParameterType": "Token"
		  },
		  {
		  "Id": 29,
		  "Key": "AppOwnerFirstName",
		  "DefaultValue": "metadata:AppOwnerFirstName",
		  "ParameterType": "Token"
		  },
		  {
		  "Id": 30,
		  "Key": "Text",
		  "DefaultValue": "Pending Alert: Enterprise Web Server",
		  "ParameterType": "Value"
		  },
		  {
		  "Id": 31,
		  "Key": "ApprovalLink",
		  "DefaultValue": "apprlink",
		  "ParameterType": "Token"
		  },
		  {
		  "Id": 32,
		  "Key": "ScriptName",
		  "DefaultValue": "MyScript.ps1",
		  "ParameterType": "Script"
		  }
		]</pre>

Name	In	Description
CARequestId		A string containing the CA's reference ID for the certificate request.
CommonName		A string indicating the common name of the certificate.
LogicalName		A string indicating the logical name of the certificate authority.

Table 197: POST Alerts Pending Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the pending request alert.
DisplayName	A string indicating the display name for the pending request alert. This name appears in the pending request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate request or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nA certificate using the {template} template was requested by {requester:displayname} from {careqid} on {subdate}. The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>DN: {rdn}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td> </td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n {apprlink}\n\nThanks!\n\nYour Certificate Management Tool\n" See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail}

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
• Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the pending request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all pending certificate requests. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for all templates.</td></tr><tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr><tr><td>ForestRoot</td><td>A string indicating the forest root of the template.
Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr><tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>8</td><td>PendingLogger</td></tr>
<tr><td>9</td><td>PendingPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>and is used to reference the fully qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704). • Token This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings. • Value This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT Alerts Pending

The PUT /Alerts/Pending method is used to update a pending certificate request alert. This method returns HTTP 200 OK on a success with details about the pending certificate request alert.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/monitoring/alerts/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 198: PUT Alerts Pending Input Parameters

Name	In	Description
id	Path	An integer indicating the Keyfactor Command reference ID of the pending request alert.
DisplayName	Body	**Required**. A string indicating the display name for the pending request alert. This name appears in the pending request alerts grid in the Management Portal.
Subject	Body	**Required**. A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate request or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	Body	**Required**. A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nA certificate using the {template} template was requested by {requester:displayname} from {careqid} on {subdate}. The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:ApOwnerFirstName}</td></tr>\n<tr><td>DN: {rdn}</td><td>App Owner Last Name: {metadata:ApOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:ApOwnerEmailAddress}</td></tr>\n<tr><td> </td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n{apprlink}\n\nThanks!\n\nYour Certificate Management Tool\n" See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings. **Note:** The $(requester:givenname) substitutable special text

Name	In	Description
		token is only supported in environments using Active Directory as an identity provider.
Recipients	Body	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail} The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.
TemplateId	Body	An integer indicating the certificate template for which the pending request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all pending certificate requests. Use the GET /Templates method (see GET Templates on page 2422) to retrieve a list of all the templates to determine the template ID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.

<table><tr><th>ID</th><th>Event Handler Type</th></tr><tr><td>8</td><td>PendingLogger</td></tr><tr><td>9</td><td>PendingPowershell</td></tr></table></td></tr><tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr></table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr><tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr><tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr><tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging</td></tr></table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>handler and is used to reference the fully qualified domain name of the target machine to which event should be logged.
		• Script
		This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
		• Token
		This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings.
		• Value
		This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr></table>
		For example, for a PowerShell handler:
		<pre>"EventHandlerParameters": [
		  {
		  "Id": 28,
		  "Key": "cn",
		  "DefaultValue": "rcn",
		  "ParameterType": "Token"
		  },
		  {
		  "Id": 29,
		  "Key": "AppOwnerFirstName",</pre>

Name	In	Description

		"DefaultValue": "metadata:AppOwnerFirstName",
		"ParameterType": "Token"
		},
		{
		"Id": 30,
		"Key": "Text",
		"DefaultValue": "Pending Alert: Enterprise Web Server",
		"ParameterType": "Value"
		},
		{
		"Id": 31,
		"Key": "ApprovalLink",
		"DefaultValue": "apprlink",
		"ParameterType": "Token"
		},
		{
		"Id": 32,
		"Key": "ScriptName",
		"DefaultValue": "MyScript.ps1",
		"ParameterType": "Script"
		}
		]

Table 199: PUT Alerts Pending Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the pending request alert.
DisplayName	A string indicating the display name for the pending request alert. This name appears in the pending request alerts grid in the Management Portal.
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate request or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML. For example: "Hello,\n\nA certificate using the {template} template was requested by {requester:displayname} from {careqid} on {subdate}. The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:AppOwnerFirstName}</td></tr>\n<tr><td>DN: {rdn}</td><td>App Owner Last Name: {metadata:AppOwnerLastName}</td></tr>\n<tr><td>SANs: {san}</td><td>App Owner Email Address: {metadata:AppOwnerEmailAddress}</td></tr>\n<tr><td> </td><td>Business Critical: {metadata:BusinessCritical}</td></tr>\n</table>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n {apprlink}\n\nThanks!\n\nYour Certificate Management Tool\n" See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings.
Recipients	An array of strings containing the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time. Available email substitutable special text strings include: • {requester:mail}

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
• Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>Template</td><td>An object containing information about the certificate template for which the pending request alerts will be generated. A separate alert should be configured for each template. An alert may be configured with no template, if desired. Alerts configured in this way generate alerts for all pending certificate requests. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the template, or null for all templates.</td></tr><tr><td>DisplayName</td><td>A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name.</td></tr><tr><td>ForestRoot</td><td>A string indicating the forest root of the template.
Note: This field is retained for legacy purposes and will be replaced by ConfigurationTenant field.</td></tr><tr><td>ConfigurationTenant</td><td>A string indicating the configuration tenant of the template.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>8</td><td>PendingLogger</td></tr>
<tr><td>9</td><td>PendingPowershell</td></tr>
</table></td></tr>
<tr><td>DisplayName</td><td>A string containing the name of the event handler.</td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers on page 218.</td></tr>
<tr><td>EventHandlerParameters</td><td>An array of objects containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>and is used to reference the fully qualified domain name of the target machine to which event should be logged.
• Script
This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the script name as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
• Token
This type is used for the PowerShell handler and is used to reference a substitutable special text value that should be passed to the PowerShell script. See Table 5: Substitutable Special Text for Pending Request Alerts for a complete list of available substitutable special text strings.
• Value
This type is used for the PowerShell handler and is used to reference a static text string that should be passed to the PowerShell script.</td></tr>
</table>
</td></tr>
<tr><td>CARequestId</td><td>A string containing the CA's reference ID for the certificate request.</td></tr>
<tr><td>CommonName</td><td>A string indicating the common name of the certificate.</td></tr>
<tr><td>LogicalName</td><td>A string indicating the logical name of the certificate authority.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Pending Test

The POST /Alerts/Pending/Test method is used to test individual pending certificate request alerts. This method returns HTTP 200 OK on a success with details about the resulting alerts generated.

Tip: Alerts are generated for all certificate requests that have not previously been alerted on, unless the system has been configured to send multiple alerts per request. By default, one alert is sent to each recipient for any given request. The number of alerts to send for a given request is configurable with the *Pending Alert Max Reminders* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If a certificate remains in a pending state after the configured number of alerts has been sent, no further alerts will be sent.
By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Pending Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If more than 100 alerts are generated, no email messages will be sent and you'll have the opportunity to view the first 100 alerts generated.

If you're using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message) when the test is run. This is true regardless of the setting of the *sendAlertsEmails* flag.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 200: POST Alerts Pending Test Input Parameters

Name	In	Description
AlertId	Body	An integer indicating the Keyfactor Command reference ID for the pending alert.
SendAlerts	Body	A Boolean indicating whether to send alert emails with the test (true), or not (false).

Table 201: POST Alerts Pending Test Response Data

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>PendingAlerts</td><td>An object containing alert details resulting from the test. Pending alert details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Subject</td><td>A string indicating the subject for the email message that will be delivered when the alert is triggered.

Tip: Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate request or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.</td></tr>
<tr><td>Message</td><td>A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML.</td></tr>
<tr><td>Recipients</td><td>An array of strings containing a list of strings with the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time.</td></tr>
<tr><td>CARequestId</td><td>An string containing the CA’s reference ID for the certificate request.</td></tr>
<tr><td>CommonName</td><td>A string indicating the common name of the certificate request.</td></tr>
<tr><td>LogicalName</td><td>A string indicating the logical name of the certificate authority from which the certificate was requested.</td></tr>
</table></td></tr>
<tr><td>AlertBuildResult</td><td>A string indicating the result of pending alerts test (e.g. Success).</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST Alerts Pending TestAll

The POST /Alerts/Pending/TestAll method is used to test all pending certificate request alerts. This method returns HTTP 200 OK on a success with details about the resulting number of alerts generated.

Tip: Alerts are generated for all certificate requests that have not previously been alerted on, unless the system has been configured to send multiple alerts per request. By default, one alert is sent to each recipient for any given request. The number of alerts to send for a given request is configurable with the *Pending Alert Max Reminders* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If a certificate remains in a pending state after the configured number of alerts has been sent, no further alerts will be sent.
By default, a maximum of 100 alerts will be generated during a test. The maximum value is configurable with the *Pending Alert Test Result Limit* setting in Keyfactor Command application settings (see Application Settings: Console Tab on page 602). If more than 100 alerts are generated, no email messages will be sent and you'll have the opportunity to view the first 100 alerts generated.

If you're using an event handler, the event handler is run and the handler actions taken (PowerShell script run, event log message) when the test is run. This is true regardless of the setting of the *sendAlertsEmails* flag.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 202: POST Alerts Pending Test All Input Parameters

Name	In	Description
SendAlerts	Body	A Boolean indicating whether to send alert emails with the test (true), or not (false).

Table 203: POST Alerts Pending Test All Response Data

Name	Description
PendingAlerts	An object containing alert details resulting from the test. Pending alert details are:

Name	Description
Subject	A string indicating the subject for the email message that will be delivered when the alert is triggered. **Tip:** Substitutable special text may be used in the subject line. Substitutable special text uses a variable in the alert definition that is replaced by data from the certificate request or certificate metadata at processing time. For example, you can enter {rcn} in the alert definition and each alert generated at processing time will contain the specific requested common name of the given certificate request instead of the variable {rcn}.
Message	A string indicating the email message that will be delivered when the alert is triggered. The email message is made up of regular text and substitutable special text. If desired, you can format the message body using HTML.
Recipients	An array of strings containing a list of strings with the recipients for the alert. Each alert can have multiple recipients. You can use specific email addresses and/or use substitutable special text to replace an email address variable with actual email addresses at processing time.
CARequestId	An string containing the CA's reference ID for the certificate request.
CommonName	A string indicating the common name of the certificate request.
LogicalName	A string indicating the logical name of the certificate authority from which the certificate was requested.

AlertBuildResult	An integer indicating the number of pending alerts run by the test.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.5 AppSetting

The AppSetting component of the Keyfactor API includes methods necessary to list and update application settings that control the behavior of Keyfactor Command features. For a complete list of available application settings, see Application Settings on page 601 in the *Keyfactor Command Reference Guide*.

Table 204: AppSetting Endpoints

Endpoint	Method	Description	Link
/	GET	Returns details for all the application settings.	GET AppSetting below
/	PUT	Updates values configured for multiple application settings in a single command.	PUT AppSetting on page 1079
/{id}	GET	Returns details for a single application setting.	GET AppSetting ID on page 1077
/{id}/Set	PUT	Updates the value configured for an application setting based on its reference ID.	PUT AppSetting ID Set on page 1081
/{name}/Set	PUT	Updates the value configured for an application setting based on its reference name.	PUT AppSetting Name Set on page 1083

3.6.5.1 GET AppSetting

The GET /AppSetting method is used to retrieve the details for all the application settings in Keyfactor Command. This method returns HTTP 200 OK on a success with a list of the application setting details. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/application_settings/read/

Table 205: GET AppSetting Response Data

Name	Description
Id	Integer indicating the Keyfactor Command reference ID of the application setting.
DisplayName	A string indicating the name for the application setting in the Keyfactor Command Management Portal.
ShortName	A string indicating the Keyfactor Command internal reference name for the application setting.
Description	A string indicating the description for the application setting. This description appears in the Keyfactor Command Management Portal when you hover over the *DisplayName* for the application setting.
Value	A field indicating the value for the application setting. May be a Boolean, integer, or string.
ValueType	An integer indicating the type for the *Value*. Possible value types are:

Value	Parameter Value
0	String
1	Integer
2	Boolean
4	String (RegEx)
5	String (URL)
6	String (Path)
7	String (CA Name)
8	No longer in use
9	String (Template Name)
10	String (Date)

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.5.2 GET AppSetting ID

The GET /AppSetting/{id} method is used to retrieve a single application setting from Keyfactor Command. This method returns HTTP 200 OK on a success with a list of the application setting details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/application_settings/read/

Table 206: GET AppSetting {id} Input Parameters

Name	In	Description
id	Path	**Required**. Integer indicating the Keyfactor Command reference ID of the application setting to retrieve. Use the *GET /AppSetting* method (see GET Agents on page 858) to retrieve a list of all the application settings to determine the application setting ID.

Table 207: GET AppSetting {id} Response Data

Name	Description
Id	Integer indicating the Keyfactor Command reference ID of the application setting.
DisplayName	A string indicating the name for the application setting in the Keyfactor Command Management Portal.
ShortName	A string indicating the Keyfactor Command internal reference name for the application setting.
Description	A string indicating the description for the application setting. This description appears in the Keyfactor Command Management Portal when you hover over the *DisplayName* for the application setting.
Value	A field indicating the value for the application setting. May be a Boolean, integer, or string.
ValueType	An integer indicating the type for the *Value*. Possible value types are:

Value	Parameter Value
0	String
1	Integer
2	Boolean
4	String (RegEx)
5	String (URL)
6	String (Path)
7	String (CA Name)
8	No longer in use
9	String (Template Name)
10	String (Date)

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.5.3 PUT AppSetting

The PUT /AppSetting method is used to update the values of multiple application settings with a single command. This method returns HTTP 200 OK on a success with information about the updated application settings.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/application_settings/read/
/application_settings/modify/

Table 208: PUT AppSetting Input Parameters

Name	In	Description
Id	Body	**Required**. Integer indicating the Keyfactor Command reference ID of the application setting. Use the GET /AppSetting method (see GET AppSetting on page 1075) to retrieve a list of all the application settings to determine the application setting ID.
Value	Body	**Required**. A field indicating the value for the application setting. May be a Boolean, integer, or string.

Table 209: PUT AppSetting Response Data

Name	Description
Id	Integer indicating the Keyfactor Command reference ID of the application setting.
DisplayName	A string indicating the name for the application setting in the Keyfactor Command Management Portal.
ShortName	A string indicating the Keyfactor Command internal reference name for the application setting.
Description	A string indicating the description for the application setting. This description appears in the Keyfactor Command Management Portal when you hover over the *DisplayName* for the application setting.
Value	A field indicating the value for the application setting. May be a Boolean, integer, or string.
ValueType	An integer indicating the type for the *Value*. Possible value types are:

Value	Parameter Value
0	String
1	Integer
2	Boolean
4	String (RegEx)
5	String (URL)
6	String (Path)
7	String (CA Name)
8	No longer in use
9	String (Template Name)
10	String (Date)

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.5.4 PUT AppSetting ID Set

The PUT /AppSetting/{id}/Set method is used to update the value of an application setting specified by the reference ID. This method returns HTTP 200 OK on a success with information about the updated application setting.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/application_settings/read/
/application_settings/modify/

Table 210: PUT AppSetting {id} Set Input Parameters

Name	In	Description
id	Path	**Required**. Integer indicating the Keyfactor Command reference ID of the application setting. Use the GET /AppSetting method (see GET AppSetting on page 1075) to retrieve a list of all the application settings to determine the application setting ID.
Value	Body	**Required**. A field indicating the value for the application setting. May be a Boolean, integer, or string.

Table 211: PUT AppSetting {id} Set Response Data

Name	Description
Id	Integer indicating the Keyfactor Command reference ID of the application setting.
DisplayName	A string indicating the name for the application setting in the Keyfactor Command Management Portal.
ShortName	A string indicating the Keyfactor Command internal reference name for the application setting.
Description	A string indicating the description for the application setting. This description appears in the Keyfactor Command Management Portal when you hover over the *DisplayName* for the application setting.
Value	A field indicating the value for the application setting. May be a Boolean, integer, or string.
ValueType	An integer indicating the type for the *Value*. Possible value types are:

Value	Parameter Value
0	String
1	Integer
2	Boolean
4	String (RegEx)
5	String (URL)
6	String (Path)
7	String (CA Name)
8	No longer in use
9	String (Template Name)
10	String (Date)

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.5.5 PUT AppSetting Name Set

The PUT /AppSetting/{name}/Set method is used to update the value of an application setting specified by the reference name. This method returns HTTP 200 OK on a success with information about the updated application settings.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/application_settings/read/
/application_settings/modify/

Table 212: PUT AppSetting {name} Set Input Parameters

Name	In	Description
name	Path	**Required**. A string indicating the Keyfactor Command internal reference name (*ShortName*) for the application setting. Use the GET /AppSetting method (see GET AppSetting on page 1075) to retrieve a list of all the application settings to determine the application setting reference name (*ShortName*).
Value	Body	**Required**. A field indicating the value for the application setting. May be a Boolean, integer, or string.

Table 213: PUT AppSetting {name} Set Response Data

Name	Description
Id	Integer indicating the Keyfactor Command reference ID of the application setting.
DisplayName	A string indicating the name for the application setting in the Keyfactor Command Management Portal.
ShortName	A string indicating the Keyfactor Command internal reference name for the application setting.
Description	A string indicating the description for the application setting. This description appears in the Keyfactor Command Management Portal when you hover over the *DisplayName* for the application setting.
Value	A field indicating the value for the application setting. May be a Boolean, integer, or string.
ValueType	An integer indicating the type for the *Value*. Possible value types are:

Value	Parameter Value
0	String
1	Integer
2	Boolean
4	String (RegEx)
5	String (URL)
6	String (Path)
7	String (CA Name)
8	No longer in use
9	String (Template Name)
10	String (Date)

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.6 Audit

The Audit component of the Keyfactor API is used to track changes to the Keyfactor Command operation and configuration.

Table 214: Audit Endpoints

Endpoint	Method	Description	Links
/{id}	GET	Returns information about the specified audit log entry.	GET Audit ID below
/{id}/Validate	GET	Validates the specified audit log entry.	GET Audit ID Validate on page 1090
/	GET	Returns a list of all audit log entries according to the provided filters and input parameters.	GET Audit on page 1091
/Download	GET	Returns a comma separated list of audit log entries according to the provided filters and input parameters.	GET Audit Download on page 1097
/RelatedEntities	GET	Returns a list of all audit log entries and entries related to this entry according to the provided filters and input parameters.	GET Audit Related Entities on page 1101

3.6.6.1 GET Audit ID

The GET /Audit/{id} method is used to retrieve details for a specified audit entry. This method returns HTTP 200 OK on a success with audit log details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/auditing/read/

Table 215: GET Audit {id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the audit log entry to retrieve. Use the *GET /Audit* method (see GET Audit on page 1091) to retrieve a list of all the audit log entries to determine the audit log entry ID.

Table 216: GET Audit {id} Response Data

Name	Description
Id	The ID of the specified audit log entry.
TimeStamp	The timestamp (UTC) on the audit log entry indicating when the action performed occurred.
Message	XML data on the audit event.
Signature	The signature on the audit entry.
Category	An integer identifying the category of the audit entry. Possible values are:

Value	Subcategory Name	Description
2001	Certificate	Certificate
2001	Audit-ingCertificateScheduledReplacement	Auditing Certi-ficate Scheduled Replacement
2001	AuditingCertificateRequest	Certificate Request
2002	ApiApplication	API Application
2003	Template	Template
2004	CertificateQuery	Certificate Collec-tion/Query
2005	ExpirationAlert	Expiration Alert
2005	ExpirationAlertDefinitionContextModel	Expiration Alert
2006	PendingAlert	Pending Alert
2006	PendingAlertDefinitionContextModel	Pending Alert
2007	ApplicationSetting	Application Setting
2008	IssuedAlert	Issued Alert
2008	IssuedAlertDefinitionContextModel	Issued Alert

Name	Description

Value	Subcategory Name	Description
2009	DeniedAlert	Denied Alert
2009	DeniedAlertDefinitionContextModel	Denied Alert
2010	ADIdentityModel	Security Identity
2011	SecurityRole	Security Role
2012	AuthorizationFailure	Authorization Failure
2013	CertificateSigningRequest	CSR
2014	ServerGroup	SSH Server Group
2015	Server	SSH Server
2016	DiscoveredKey	Rogue Key for Logon
2016	Key	SSH Key
2017	ServiceAccount	SSH Service Account
2018	Logon	SSH Logon
2019	SshUser	SSH User
2020	KeyRotationAlertDefinitionContextModel	SSH Key Rotation Alert
2021	CertificateStore	Certificate Store
2022	JobType	Orchestrator Job Type
2023	AgentSchedule	Orchestrator Job
2024	BulkAgentSchedule	Bulk Orchestrator Job
2025	CertificateStoreContainer	Store Container

Name	Description
	<table><tr><th>Value</th><th>Subcategory Name</th><th>Description</th></tr><tr><td>2026</td><td>Agent</td><td>Orchestrator</td></tr><tr><td>2027</td><td>RevocationMonitoring</td><td>Monitoring</td></tr><tr><td>2028</td><td>License</td><td>License</td></tr><tr><td>2029</td><td>WorkflowDefinition</td><td>Workflow Definition</td></tr><tr><td>2030</td><td>WorkflowInstance</td><td>Workflow Instance</td></tr><tr><td>2031</td><td>WorkflowInstanceSignal</td><td>Workflow Instance Signal</td></tr><tr><td>2032</td><td>IdentityProvider</td><td>Identity Provider</td></tr><tr><td>2033</td><td>RoleClaimDefinition</td><td>Claim Definition</td></tr><tr><td>2034</td><td>PermissionSet</td><td>Permission Set</td></tr></table>  **Tip:** To do a query by category, use the subcategory string. For example, the following query would return audit records for categories 2023, 2024, and 2026 since they all contain “Agent” in the subcategory: `category -contains "Agent"`
Operations	An integer identifying the operation of the audit entry. Possible values are: <table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Created</td></tr><tr><td>2</td><td>Updated</td></tr><tr><td>3</td><td>Deleted</td></tr><tr><td>4</td><td>Approved</td></tr><tr><td>5</td><td>Denied</td></tr><tr><td>6</td><td>Revoked</td></tr></table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>7</td><td>Downloaded</td></tr><tr><td>8</td><td>Deleted Private Key</td></tr><tr><td>9</td><td>Renewed</td></tr><tr><td>10</td><td>Encountered</td></tr><tr><td>11</td><td>Scheduled Replacement</td></tr><tr><td>12</td><td>Recovered</td></tr><tr><td>13</td><td>Imported</td></tr><tr><td>14</td><td>Removed from Hold</td></tr><tr><td>15</td><td>Scheduled Add</td></tr><tr><td>16</td><td>Scheduled Removal</td></tr><tr><td>17</td><td>Download with Private Key</td></tr><tr><td>18</td><td>Scheduled</td></tr><tr><td>19</td><td>Reset</td></tr><tr><td>20</td><td>Disapproved</td></tr><tr><td>21</td><td>Restarted</td></tr><tr><td>22</td><td>Sent</td></tr><tr><td>23</td><td>Failed</td></tr><tr><td>24</td><td>Completed</td></tr><tr><td>25</td><td>Rejected</td></tr></table>
Level	The alert level of the audit log entry. Possible values are:

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Information</td></tr><tr><td>1</td><td>Warning</td></tr><tr><td>2</td><td>Failure</td></tr></table>
User	The user who performed the audit event in DOMAIN\username format.
EntityType	The category of the object being audited (e.g. Template, Certificate).
AuditIdentifier	An identifier of the object being audited (e.g. the template name for a template, the CN for a certificate). It is important to note that this is a value that is typically used for easy identification of an object, but is not necessarily unique, and is subject to change.
ImmutableIdentifier	The fixed ID of the auditable event in the Keyfactor database.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.6.2 GET Audit ID Validate

The GET /Audit/{id}/Validate method is used to return whether or not (true or false) the audit log entry is valid. An audit log might become invalidated if it is tampered with. This method returns HTTP 200 OK on a success with a value of true or false.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/auditing/read/

Table 217: GET Audit {id} Validate Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the audit log entry to validate. Use the *GET /Audit* method (see GET Audit on the next page) to retrieve a list of all the audit log entries to determine the audit log entry ID.

Table 218: GET Audit {id} Validate Response Data

Name	Description
	A Boolean that indicates whether the audit log entry is valid (true) or not (false). This value is returned without a parameter name.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.6.3 GET Audit

The GET /Audit method returns a list of all audit entries. This method returns HTTP 200 OK on a success with audit log details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/auditing/read/

Table 219: GET Audit Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Audit Log Search Feature on page 717. The query fields supported for this endpoint are: • Name (EntityIdentifier) • Category (EntityType) (see Table 220: GET Audit Response Data for codes) • ImmutableIdentifier • Level (see Table 220: GET Audit Response Data for codes) • Operation (see Table 220: GET Audit Response Data for codes) • PropertyChanged • Timestamp • ActingUser **Tip:** To do a query by category, use the subcategory string (see *Category* in the response data). For example: `category -contains "Agent"`
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 220: GET Audit Response Data

Name	Description
Id	The ID of the specified audit log entry.
TimeStamp	The timestamp (UTC) on the audit log entry indicating when the action performed occurred.
Message	XML data on the audit event.
Signature	The signature on the audit entry.
Category	An integer identifying the category of the audit entry. Possible values are:

Value	Subcategory Name	Description
2001	Certificate	Certificate
2001	Audit-ingCertificateScheduledReplacement	Auditing Certi-ficate Scheduled Replacement
2001	AuditingCertificateRequest	Certificate Request
2002	ApiApplication	API Application
2003	Template	Template
2004	CertificateQuery	Certificate Collec-tion/Query
2005	ExpirationAlert	Expiration Alert
2005	ExpirationAlertDefinitionContextModel	Expiration Alert
2006	PendingAlert	Pending Alert
2006	PendingAlertDefinitionContextModel	Pending Alert
2007	ApplicationSetting	Application Setting
2008	IssuedAlert	Issued Alert
2008	IssuedAlertDefinitionContextModel	Issued Alert

Name	Description

Value	Subcategory Name	Description
2009	DeniedAlert	Denied Alert
2009	DeniedAlertDefinitionContextModel	Denied Alert
2010	ADIdentityModel	Security Identity
2011	SecurityRole	Security Role
2012	AuthorizationFailure	Authorization Failure
2013	CertificateSigningRequest	CSR
2014	ServerGroup	SSH Server Group
2015	Server	SSH Server
2016	DiscoveredKey	Rogue Key for Logon
2016	Key	SSH Key
2017	ServiceAccount	SSH Service Account
2018	Logon	SSH Logon
2019	SshUser	SSH User
2020	KeyRotationAlertDefinitionContextModel	SSH Key Rotation Alert
2021	CertificateStore	Certificate Store
2022	JobType	Orchestrator Job Type
2023	AgentSchedule	Orchestrator Job
2024	BulkAgentSchedule	Bulk Orchestrator Job
2025	CertificateStoreContainer	Store Container

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Subcategory Name</th><th>Description</th></tr>
<tr><td>2026</td><td>Agent</td><td>Orchestrator</td></tr>
<tr><td>2027</td><td>RevocationMonitoring</td><td>Monitoring</td></tr>
<tr><td>2028</td><td>License</td><td>License</td></tr>
<tr><td>2029</td><td>WorkflowDefinition</td><td>Workflow Definition</td></tr>
<tr><td>2030</td><td>WorkflowInstance</td><td>Workflow Instance</td></tr>
<tr><td>2031</td><td>WorkflowInstanceSignal</td><td>Workflow Instance Signal</td></tr>
<tr><td>2032</td><td>IdentityProvider</td><td>Identity Provider</td></tr>
<tr><td>2033</td><td>RoleClaimDefinition</td><td>Claim Definition</td></tr>
<tr><td>2034</td><td>PermissionSet</td><td>Permission Set</td></tr>
</table>
Tip: To do a query by category, use the subcategory string. For example, the following query would return audit records for categories 2023, 2024, and 2026 since they all contain “Agent” in the subcategory:
<pre>category -contains "Agent"</pre>
</td></tr>
<tr><td>Operations</td><td>An integer identifying the operation of the audit entry. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>Created</td></tr>
<tr><td>2</td><td>Updated</td></tr>
<tr><td>3</td><td>Deleted</td></tr>
<tr><td>4</td><td>Approved</td></tr>
<tr><td>5</td><td>Denied</td></tr>
<tr><td>6</td><td>Revoked</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>7</td><td>Downloaded</td></tr><tr><td>8</td><td>Deleted Private Key</td></tr><tr><td>9</td><td>Renewed</td></tr><tr><td>10</td><td>Encountered</td></tr><tr><td>11</td><td>Scheduled Replacement</td></tr><tr><td>12</td><td>Recovered</td></tr><tr><td>13</td><td>Imported</td></tr><tr><td>14</td><td>Removed from Hold</td></tr><tr><td>15</td><td>Scheduled Add</td></tr><tr><td>16</td><td>Scheduled Removal</td></tr><tr><td>17</td><td>Download with Private Key</td></tr><tr><td>18</td><td>Scheduled</td></tr><tr><td>19</td><td>Reset</td></tr><tr><td>20</td><td>Disapproved</td></tr><tr><td>21</td><td>Restarted</td></tr><tr><td>22</td><td>Sent</td></tr><tr><td>23</td><td>Failed</td></tr><tr><td>24</td><td>Completed</td></tr><tr><td>25</td><td>Rejected</td></tr></table>
Level	The alert level of the audit log entry. Possible values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Information</td></tr>
<tr><td>1</td><td>Warning</td></tr>
<tr><td>2</td><td>Failure</td></tr>
</table>
</td></tr>
<tr><td>User</td><td>The user who performed the audit event in DOMAIN\username format.</td></tr>
<tr><td>EntityType</td><td>The category of the object being audited (e.g. Template, Certificate).</td></tr>
<tr><td>AuditIdentifier</td><td>An identifier of the object being audited (e.g. the template name for a template, the CN for a certificate). It is important to note that this is a value that is typically used for easy identification of an object, but is not necessarily unique, and is subject to change.</td></tr>
<tr><td>ImmutableIdentifier</td><td>The fixed ID of the auditable event in the Keyfactor database.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.6.4 GET Audit Download

The GET /Audit/Download method returns a comma-delimited list of all audit entries matching the requested filters appropriate for output to a CSV file. This method returns HTTP 200 OK on a success with the information requested in comma-delimited form with the property names at the start of the list and then the values.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/auditing/read/

Table 221: GET Audit Download Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Audit Log Search Feature on page 717. The query fields supported for this endpoint are: • Name (EntityIdentifier) • Category (EntityType) (see Table 220: GET Audit Response Data for codes) • ImmutableIdentifier • Level (see Table 220: GET Audit Response Data for codes) • Operation (see Table 220: GET Audit Response Data for codes) • PropertyChanged • Timestamp • ActingUser
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 222: GET Audit Download Response Data

Name	Description
Id	The ID of the specified audit log entry.
TimeStamp	The timestamp (UTC) on the audit log entry indicating when the action performed occurred.
Message	The message as displayed in the Keyfactor Command Management Portal.
Message	XML data on the audit event. Also known as the *XMLMessage* in some interfaces.
Operations	An integer identifying the operation of the audit entry. Possible values are:

Value	Description
1	Created
2	Updated
3	Deleted
4	Approved
5	Denied
6	Revoked
7	Downloaded
8	Deleted Private Key
9	Renewed
10	Encountered
11	Scheduled Replacement
12	Recovered
13	Imported
14	Removed from Hold
15	Scheduled Add
16	Scheduled Removal

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>17</td><td>Download with Private Key</td></tr>
<tr><td>18</td><td>Scheduled</td></tr>
<tr><td>19</td><td>Reset</td></tr>
<tr><td>20</td><td>Disapproved</td></tr>
<tr><td>21</td><td>Restarted</td></tr>
<tr><td>22</td><td>Sent</td></tr>
<tr><td>23</td><td>Failed</td></tr>
<tr><td>24</td><td>Completed</td></tr>
<tr><td>25</td><td>Rejected</td></tr>
</table>
</td></tr>
<tr><td>Level</td><td>The alert level of the audit log entry. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Information</td></tr>
<tr><td>1</td><td>Warning</td></tr>
<tr><td>2</td><td>Failure</td></tr>
</table>
</td></tr>
<tr><td>User</td><td>The user who performed the audit event in DOMAIN\username format.</td></tr>
<tr><td>EntityType</td><td>The category of the object being audited (e.g. Template, Certificate). Also known as the Category in some interfaces.</td></tr>
<tr><td>AuditIdentifier</td><td>An identifier of the object being audited (e.g. the template name for a template, the CN for a certificate). It is important to note that this is a value that is typically used for easy identification of an object, but is not necessarily unique, and is subject to change. Also known as the Name in some interfaces.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.6.5 GET Audit Related Entities

The GET /Audit/RelatedEntities method returns a list of all audit entries and all audit entries related to those audit entries. This method returns HTTP 200 OK on a success with the information requested.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/auditing/read/

Table 223: GET Audit Related Entities Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Audit Log Search Feature on page 717. The query fields supported for this endpoint are: • Name (EntityIdentifier) • Category (EntityType) (see Table 220: GET Audit Response Data for codes) • ImmutableIdentifier • Level (see Table 220: GET Audit Response Data for codes) • Operation (see Table 220: GET Audit Response Data for codes) • PropertyChanged • Timestamp • ActingUser **Tip:** In order to return related entries, your queryString needs to query for the specific immutable identifier of the audit record for which you wish to see related entries. For example: `ImmutableIdentifier -eq 707662`
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 224: GET Audit Related Entities Response Data

Name	Description
Id	The ID of the specified audit log entry.
TimeStamp	The timestamp (UTC) on the audit log entry indicating when the action performed occurred.
Message	XML data on the audit event.
Signature	The signature on the audit entry.
Category	An integer identifying the category of the audit entry. Possible values are:

Value	Subcategory Name	Description
2001	Certificate	Certificate
2001	Audit-ingCertificateScheduledReplacement	Auditing Certi-ficate Scheduled Replacement
2001	AuditingCertificateRequest	Certificate Request
2002	ApiApplication	API Application
2003	Template	Template
2004	CertificateQuery	Certificate Collec-tion/Query
2005	ExpirationAlert	Expiration Alert
2005	ExpirationAlertDefinitionContextModel	Expiration Alert
2006	PendingAlert	Pending Alert
2006	PendingAlertDefinitionContextModel	Pending Alert
2007	ApplicationSetting	Application Setting
2008	IssuedAlert	Issued Alert
2008	IssuedAlertDefinitionContextModel	Issued Alert

Name	Description

Value	Subcategory Name	Description
2009	DeniedAlert	Denied Alert
2009	DeniedAlertDefinitionContextModel	Denied Alert
2010	ADIdentityModel	Security Identity
2011	SecurityRole	Security Role
2012	AuthorizationFailure	Authorization Failure
2013	CertificateSigningRequest	CSR
2014	ServerGroup	SSH Server Group
2015	Server	SSH Server
2016	DiscoveredKey	Rogue Key for Logon
2016	Key	SSH Key
2017	ServiceAccount	SSH Service Account
2018	Logon	SSH Logon
2019	SshUser	SSH User
2020	KeyRotationAlertDefinitionContextModel	SSH Key Rotation Alert
2021	CertificateStore	Certificate Store
2022	JobType	Orchestrator Job Type
2023	AgentSchedule	Orchestrator Job
2024	BulkAgentSchedule	Bulk Orchestrator Job
2025	CertificateStoreContainer	Store Container

Name	Description
	<table><tr><th>Value</th><th>Subcategory Name</th><th>Description</th></tr><tr><td>2026</td><td>Agent</td><td>Orchestrator</td></tr><tr><td>2027</td><td>RevocationMonitoring</td><td>Monitoring</td></tr><tr><td>2028</td><td>License</td><td>License</td></tr><tr><td>2029</td><td>WorkflowDefinition</td><td>Workflow Definition</td></tr><tr><td>2030</td><td>WorkflowInstance</td><td>Workflow Instance</td></tr><tr><td>2031</td><td>WorkflowInstanceSignal</td><td>Workflow Instance Signal</td></tr><tr><td>2032</td><td>IdentityProvider</td><td>Identity Provider</td></tr><tr><td>2033</td><td>RoleClaimDefinition</td><td>Claim Definition</td></tr><tr><td>2034</td><td>PermissionSet</td><td>Permission Set</td></tr></table>  **Tip:** To do a query by category, use the subcategory string. For example, the following query would return audit records for categories 2023, 2024, and 2026 since they all contain “Agent” in the subcategory: `category -contains "Agent"`
Operations	An integer identifying the operation of the audit entry. Possible values are: <table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Created</td></tr><tr><td>2</td><td>Updated</td></tr><tr><td>3</td><td>Deleted</td></tr><tr><td>4</td><td>Approved</td></tr><tr><td>5</td><td>Denied</td></tr><tr><td>6</td><td>Revoked</td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>7</td><td>Downloaded</td></tr>
<tr><td>8</td><td>Deleted Private Key</td></tr>
<tr><td>9</td><td>Renewed</td></tr>
<tr><td>10</td><td>Encountered</td></tr>
<tr><td>11</td><td>Scheduled Replacement</td></tr>
<tr><td>12</td><td>Recovered</td></tr>
<tr><td>13</td><td>Imported</td></tr>
<tr><td>14</td><td>Removed from Hold</td></tr>
<tr><td>15</td><td>Scheduled Add</td></tr>
<tr><td>16</td><td>Scheduled Removal</td></tr>
<tr><td>17</td><td>Download with Private Key</td></tr>
<tr><td>18</td><td>Scheduled</td></tr>
<tr><td>19</td><td>Reset</td></tr>
<tr><td>20</td><td>Disapproved</td></tr>
<tr><td>21</td><td>Restarted</td></tr>
<tr><td>22</td><td>Sent</td></tr>
<tr><td>23</td><td>Failed</td></tr>
<tr><td>24</td><td>Completed</td></tr>
<tr><td>25</td><td>Rejected</td></tr>
</table>
</td></tr>
<tr><td>Level</td><td>The alert level of the audit log entry. Possible values are:</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Information</td></tr><tr><td>1</td><td>Warning</td></tr><tr><td>2</td><td>Failure</td></tr></table>
User	The user who performed the audit event in DOMAIN\username format.
EntityType	The category of the object being audited (e.g. Template, Certificate).
AuditIdentifier	An identifier of the object being audited (e.g. the template name for a template, the CN for a certificate). It is important to note that this is a value that is typically used for easy identification of an object, but is not necessarily unique, and is subject to change.
ImmutableIdentifier	The fixed ID of the auditable event in the Keyfactor database.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7 Certificates

The Certificates component of the Keyfactor API supports certificate lifecycle and management tasks, apart from enrollment.

Table 225: Certificates Endpoints

Endpoint	Method	Description	Link
/{id}/Security	GET	Returns details of the security identities that have been granted permissions to the specified certificate including what the specific permissions are.	GET Certificates ID Security on page 1109
/{id}/Validate	GET	Validates that a certificate chain can be built for the specified certificate.	GET Certificates ID Validate on page 1111

Endpoint	Method	Description	Link
/Locations/{id}	GET	Returns details about the certificates stores in which the certificate is located.	GET Certificates Locations ID on page 1116
/IdentityAudit/{id}	GET	Returns audit identity permissions for certificate.	GET Certificates Identity Audit ID on page 1119
/{id}	DELETE	Deletes a certificate from the Keyfactor Command database by its ID.	DELETE Certificates ID on page 1123
/{id}	GET	Returns certificate details for a specified certificate.	GET Certificates ID on page 1124
/Metadata/Compare	GET	Compares the metadata value provided with the metadata value associated with the specified certificate.	GET Certificates Metadata Compare on page 1136
/{id}/History	GET	Returns the certificate operations history for a specified certificate.	GET Certificates ID History on page 1138
/	DELETE	Deletes multiple certificates from the Keyfactor Command database, as specified by the IDs in the request body.	DELETE Certificates on page 1140
/	GET	Returns all certificates with paging (number of pages to return and number of results per page) and verbosity option to specify detail level.	GET Certificates on page 1141
/Metadata	PUT	Updates the metadata for a specified certificate.	PUT Certificates Metadata on page 1156
/Metadata/All	PUT	Updates the metadata for an array of certificate IDs.	PUT Certificates Metadata All on page 1158
/Import	POST	Imports a certificate into Keyfactor Command.	POST Certificates Import on page 1162
/Revoke	POST	Revokes a certificate.	POST Certificates

Endpoint	Method	Description	Link
			Revoke on page 1166
/Analyze	POST	Reads a base-64 encoded PEM certificates and returns it in human-readable form.	POST Certificates Analyze on page 1168
/Recover	POST	Returns a recovered certificate as a PFX.	POST Certificates Recover on page 1169
/Download	POST	Downloads a certificate.	POST Certificates Download on page 1173
/RevokeAll	POST	Revokes all the certificates in the provided query.	POST Certificates Revoke All on page 1176
/Query	DELETE	Deletes multiple certificates from the Keyfactor Command database based on search query.	DELETE Certificates Query on page 1179
/PrivateKey	DELETE	Deletes the stored private keys of multiple certificates within the Keyfactor Command database.	DELETE Certificates Private Key on page 1180
/PrivateKey/{id}	DELETE	Deletes the stored private key(s) of a certificate within the Keyfactor Command database.	DELETE Certificates Private Key ID on page 1181

3.6.7.1 GET Certificates ID Security

The GET /Certificates/{id}/Security method is used to return details of permission granted to a specific certificate with the specified ID. This method returns HTTP 200 OK on a success with security details in the message body. Both global and collection-level permissions are included in the response.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
/security/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
/security/read/
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 226: GET Certificates {id} Security Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the certificate for which to check security permissions.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Table 227: GET Certificates {id} Security Response Data

Name	Description
Roles	An array of objects containing the certificate-specific permissions granted to the named security identity broken down by permission and defined by role. All roles are returned, including those that have no permissions. Role information includes:

Name	Description
Name	A string containing the short reference name for the security role.
Permissions	An array of strings containing the permissions assigned to the role.

For example, the following return snippet shows the response for the *Power Users* security role:

```
{
   "Name": "Power Users",
   "Permissions": [
      "Read",
      "EditMetadata",
      "Recover"
   ]
}
```

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.2 GET Certificates ID Validate

The GET /Certificates/{id}/Validate method is used to return details for the validity of the X509 certificate chain for the certificate with the specified ID. This method returns HTTP 200 OK on a success with certificate chain validity details in the message body.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 228: GET Certificates {id} Validate Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the certificate to be validated.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Table 229: GET Certificates {id} Validate Response Data

Name	Description
Valid	A Boolean that indicates whether all the validity tests are in a passing state (true) or not (false).
Results	An object containing the X509 certificate chain validity fields. The included validity fields are:

Name	Portal Equivalent	Description
NotTimeValid	Time Valid	A value of *Pass* indicates that the certificate time value is valid. A time can appear invalid (*Fail*) for a certificate that has expired.
NotTimeNested	n/a	A value of *Pass* indicates that the CA certificate and issued certificate have nested validity periods. A value of *Fail* can occur if the CA certificate expires before the issued certificate. This is considered deprecated and may be removed in a future release.
Revoked	Active	A value of *Pass* indicates that the X509 certificate chain is valid for the certificate and contains no revoked certificates or errors.
NotSignatureValid	Signature	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid

Name	Description

Name	Portal Equivalent	Description
		certificate signature.
NotValidForUsage	Usage	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid key usage.
UntrustedRoot	Trusted Root	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an untrusted root certificate.
RevocationStatusUnknown	Revocation Status	A value of *Pass* indicates that the revocation status can successfully be determined for the certificate. This may be the result of successful access to online certificate revocation lists (CRLs).
Cyclic	Chain Built	A value of *Pass* indicates that the certificate chain for the certificate could successfully be built.
InvalidExtension	Extensions	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid extension.
InvalidPolicyConstraints	Policy Constraints	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid policy constraint.

Name	Description

Name	Portal Equivalent	Description
InvalidBasicConstraints	Basic Constraints	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid basic constraint.
InvalidNameConstraints	Valid Name Constraints	A value of *Fail* indicates that the X509 certificate chain is invalid as a result of an invalid name constraint.
HasNotSupportedNameConstraint	Supported Name Constraints	A value of *Fail* indicates that a name constraint for the certificate is unsupported or that the certificate has no supported name constraints.
HasNotDefinedNameConstraint	Defined Name Constraints	A value of *Fail* indicates that a name constraint for the certificate is undefined.
HasNotPermittedNameConstraint	Permitted Name Constraints	A value of *Fail* indicates that a name constraint for the certificate is impermissible.
HasExcludedNameConstraint	Excluded Name Constraints	A value of *Fail* indicates that a name constraint for the certificate has been excluded.
PartialChain	Full Chain	A value of *Pass* indicates that the certificate chain for the certificate could successfully be built up to the root certificate.

Name	Description

Name	Portal Equivalent	Description
CtlNotTimeValid	CTL Time Valid	A value of *Fail* indicates that the certificate trust list (CTL) is invalid because of an invalid time value (e.g. the CTL has expired).
CtlNotSignatureValid	CTL Signature Valid	A value of *Fail* indicates that the certificate trust list (CTL) contains an invalid signature.
CtlNotValidForUsage	CTL Usage Valid	A value of *Fail* indicates that the certificate trust list (CTL) is not valid for this use.
HasWeakSignature	Strong Signature	A value of *Pass* indicates that the certificate has been signed with a secure hashing algorithm. A value of *Fail* can indicate that a hashing algorithm of MD2 or MD5 was used for the certificate.
OfflineRevocation	CRL online	A value of *Pass* indicates that the online certificate revocation list (CRL) the chain relies on is available.
NoIssuanceChainPolicy	Chain Policy	A value of *Pass* indicates that there is either no certificate policy by design in the certificate or that if a group policy has specified that all certificates must have a

Name	Description
	<table><tr><th>Name</th><th>Portal Equivalent</th><th>Description</th></tr><tr><td></td><td></td><td>certificate policy, the certificate policy exists in the certificate.</td></tr><tr><td>ExplicitDistrust</td><td>No Explicit Distrust</td><td>A value of *Pass* indicates that the certificate is not explicitly distrusted.</td></tr><tr><td>HasNotSupportedCriticalExtension</td><td>Critical Extensions</td><td>A value of *Pass* indicates that the certificate has a critical extension that is supported or has no critical extensions.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.3 GET Certificates Locations ID

The GET /Certificates/Locations/{id} method is used to return details for the certificate store locations in which the certificate with the specified ID is found. This method returns HTTP 200 OK on a success with certificate store location details in the message body.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 230: GET Certificates Locations {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the certificate for which to retrieve certificate store location details.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Table 231: GET Certificates Locations {id} Response Data

Name	Description
Details	An array of objects containing the certificate stores in which the certificate is found. Certificate store details include:

Name	Description
StoreType	A string indicating the type of certificate store (e.g. Java Keystore).
StoreTypeId	An integer indicating the Keyfactor Command referenced ID for the type of certificate store. Use the *GET CertificateStoreTypes* method (see GET Certificate Store Types on page 1546) to retrieve a list of all the certificate store types to see a complete list of types.
StoreCount	An integer indicating the number of stores of the type referenced by StoreType in which the certificate is found.
Locations	An array of objects containing details about the specific certificate stores in which the certificate is found. The following details are included about each store:

Name	Description
StoreId	A GUID that identifies the certificate store in which the certificate is located.
StoreTypeId	An integer indicating the Keyfactor Command reference ID for the type of certificate store.
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
StorePath	A string containing the path to the certi-

Name	Description
	Name / **Description**
	ficate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information.
	Alias — A string containing the alias of the certificate in the certificate store. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a user-provided string, but for an IIS Personal store, this will be the thumbprint of the certificate. See PFX Enrollment on page 146 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.4 GET Certificates Identity Audit ID

The GET /Certificates/IdentityAudit/{id} method is used to return a list of all the users or groups defined in the system that have permission to the certificate ID entered. This method returns HTTP 200 OK on a success with certificate identity audit details in the message body.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR

/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the GET /Certificates/IdentityAudit/{id} method redesigns the response to support security claims and environments with either an OAuth identity provider or Active Directory as an identity provider.

Table 232: GET Certificates Identity Audit {id} v2 Input Parameters

Name	In	Description
id	Path	**Required**. An integer containing the Keyfactor Command reference ID of the certificate.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Table 233: GET Certificates Identity Audit {id} v2 Response Data

Name	Description
Identity	An object containing information about the identity. Identity details include: **Parameter** / **Description** Id — An integer containing the Keyfactor Command reference ID for the security claim. Description — A string indicating a description for the security claim. ClaimType — A string indicating the type of claim. ClaimValue — A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$). Provider — An object containing information about the provider assigned to the security claim. **Name** / **Description** Id — A string indicating the Keyfactor Command reference GUID for the provider. Name — A string containing the short reference name for the provider (e.g. Active Directory).
Permissions	An array of objects containing the permissions granted to the certificate. **Parameter** / **Description** Name — A string containing the name of the permission (for example: Read, EditMetadata, Import, Recover, etc…)

Name	Description
	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>GrantedBy</td><td>A array of strings containing the list of roles or collections that grant the given permission to the entity.</td></tr></table>

Version 1

Version 1 of the GET /Certificates/IdentityAudit/{id} method includes the same functionality as version 2 with similar data in the response but supports only environments using Active Directory as an identity provider.

Table 234: GET Certificates Identity Audit {id} Iv1 nput Parameters

Name	In	Description
id	Path	**Required**. An integer containing the Keyfactor Command reference ID of the certificate.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Table 235: GET Certificates Identity Audit {id} v1 Response Data

Name	Description
Id	An integer containing the Keyfactor Command reference ID of the identity.
AccountName	A string containing the name of the identity.
IdentityType	A string that specifies the type of identity the entity is. For Active Directory, this will be a user or a group.
SID	A string containing the SID of the identity.
Permissions	An array of objects containing the permissions granted to the certificate. **Parameter** / **Description** Name — A string containing the name of the permission (for example: Read, EditMetadata, Import, Recover, etc…) GrantedBy — A array of strings containing the list of roles or collections that grant the given permission to the entity.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.5 DELETE Certificates ID

The DELETE /Certificates/{id} method is used to delete an existing certificate with the specified ID from the Keyfactor Command database. If the specified certificate has an associated private key stored in the database, this private key is also removed. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/delete/
OR
/certificates/collections/delte/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Tip: Deleting a certificate with this method does not necessarily delete it permanently. The certificate will be returned to the Keyfactor Command database on the next full synchronization if synchronization for the certificate source (certificate authority, SSL endpoint, etc.) is still configured. Certificate history, metadata, and private keys do not return when certificates re-synchronize. The certificate will be assigned a different Keyfactor Command reference ID when re-added to Keyfactor Command.

Table 236: DELETE Certificates {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the certificate to delete. Use the *GET /Certificates* method (see GET Certificates on page 1141) to retrieve a list of certificates based on entered search criteria to determine the certificate ID. This information is also available in the certificate details for a certificate in the Keyfactor Command Management Portal.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.6 GET Certificates ID

The GET /Certificates/{id} method is used to return details for the certificate with the specified ID. This method returns HTTP 200 OK on a success with certificate details in the message body.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR

/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 237: GET Certificates {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the certificate. Use the *GET /Certificates* method (see GET Certificates on page 1141) to retrieve a list of multiple certificates to determine the desired certificate's ID.
includeLocations	Query	A Boolean that sets whether to include the *Locations* data in the response (true) or not (false). If false is selected, the *LocationsCount* and *Locations* fields will still appear in the response, but they will contain no data. The default is *false*.
includeMetadata	Query	A Boolean that sets whether to include the *Metadata* data in the response (true) or not (false). If false is selected, the *Metadata* field will still appear in the response, but it will contain no data. The default is *false*.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Table 238: GET Certificates {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the certificate.
Thumbprint	A string indicating the thumbprint of the certificate.
SerialNumber	A string indicating the serial number of the certificate.
IssuedDN	A string indicating the distinguished name of the certificate.
IssuedCN	A string indicating the common name of the certificate.
ImportDate	The date, in UTC, on which the certificate was imported into Keyfactor Command.
NotBefore	The date, in UTC, on which the certificate was issued by the certificate authority.
NotAfter	The date, in UTC, on which the certificate expires.
IssuerDN	A string indicating the distinguished name of the issuer.
PrincipalId	An integer indicating the Keyfactor Command reference ID of the principal (UPN) that requested the certificate. Typically, this field is only populated for end user certificates requested through Keyfactor Command (e.g. Mac auto-enrollment certificates). See also *PrincipalName*.
TemplateId	An integer indicating the Keyfactor Command reference ID of the template associated with the certificate.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CertState</td><td>An integer specifying the state of the certificate. The possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Active</td></tr><tr><td>2</td><td>Revoked</td></tr><tr><td>3</td><td>Denied</td></tr><tr><td>4</td><td>Failed</td></tr><tr><td>5</td><td>Pending</td></tr><tr><td>6</td><td>Certificate Authority</td></tr><tr><td>7</td><td>Parent Certificate Authority</td></tr></table></td></tr>
<tr><td>KeySizeInBits</td><td>An integer specifying the key size in bits.</td></tr>
<tr><td>KeyType</td><td>An integer specifying the key type of the certificate. The possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>RSA</td></tr><tr><td>2</td><td>DSA</td></tr><tr><td>3</td><td>ECC</td></tr><tr><td>4</td><td>DH</td></tr></table></td></tr>
<tr><td>RequesterId</td><td>An integer indicating the Keyfactor Command reference ID of the identity that requested the certificate. See also RequesterName.</td></tr>
<tr><td>IssuedOU</td><td>A string indicating the organizational unit of the certificate.</td></tr>
<tr><td>IssuedEmail</td><td>A string indicating the email address of the certificate.</td></tr>
<tr><td>KeyUsage</td><td>An integer indicating the total key usage of the certificate. Key usage is</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>stored in Active Directory as a single value made of a combination of values. The values that make up the key usage value include:
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></tr>
<tr><td>0</td><td>None</td><td>No key usage parameters.</td></tr>
<tr><td>1</td><td>Encipherment Only</td><td>The key can be used for encryption only.</td></tr>
<tr><td>2</td><td>CRL Signing</td><td>The key can be used to sign a certificate revocation list (CRL).</td></tr>
<tr><td>4</td><td>Key Certificate Signing</td><td>The key can be used to sign certificates.</td></tr>
<tr><td>8</td><td>Key Agreement</td><td>The key can be used to determine key agreement, such as a key created using the Diffie-Hellman key agreement algorithm.</td></tr>
<tr><td>16</td><td>Data Encipherment</td><td>The key can be used for data encryption.</td></tr>
<tr><td>32</td><td>Key Encipherment</td><td>The key can be used for key encryption.</td></tr>
<tr><td>64</td><td>Nonrepudiation</td><td>The key can be used for authentication.</td></tr>
<tr><td>128</td><td>Digital Signature</td><td>The key can be used as a digital signature.</td></tr>
<tr><td>32768</td><td>Decipherment Only</td><td>The key can be used for decryption only.</td></tr>
</table>
For example, a value of 160 would represent a key usage of digital signature with key encipherment. A value of 224 would add nonrepudiation to those.</td></tr>
<tr><td>SigningAlgorithm</td><td>A string indicating the algorithm used to sign the certificate.</td></tr>
<tr><td>CertStateString</td><td>A string containing the certificate state. The possible values are:
Unknown (0)</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

- Active (1)
- Revoked (2)
- Denied (3)
- Failed (4)
- Pending (5)
- Certificate Authority (6)
- Parent Certificate Authority (7)
- External Validation (8)

</td></tr>
<tr><td>KeyTypeString</td><td>A string containing the key type description (e.g. RSA) as per the types and descriptions shown for KeyType.</td></tr>
<tr><td>RevocationEffDate</td><td>The date, in UTC, on which the certificate was revoked, if applicable.</td></tr>
<tr><td>RevocationReason</td><td>An integer indicating the reason the certificate was revoked. The possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Unspecified</td></tr>
<tr><td>1</td><td>Key Compromised</td></tr>
<tr><td>2</td><td>CA Compromised</td></tr>
<tr><td>3</td><td>Affiliation Changed</td></tr>
<tr><td>4</td><td>Superseded</td></tr>
<tr><td>5</td><td>Cessation Of Operation</td></tr>
<tr><td>6</td><td>Certificate Hold</td></tr>
<tr><td>999</td><td>Unknown</td></tr>
</table>
</td></tr>
<tr><td>RevocationComment</td><td>An internally used Keyfactor Command field.</td></tr>
<tr><td>CertificateAuthorityId</td><td>An integer indicating the Keyfactor Command reference ID of the certificate authority that issued the certificate.</td></tr>
<tr><td>CertificateAuthorityName</td><td>A string indicating the certificate authority that issued the certificate.</td></tr>
<tr><td>TemplateName</td><td>A string indicating the display name of the template that was used when issuing the certificate.</td></tr>
</table>

Name	Description
ArchivedKey	A Boolean that indicates whether the certificate has a key archived in the issuing CA (true) or not (false).
HasPrivateKey	A Boolean that indicates whether the certificate has a private key stored in Keyfactor Command (true) or not (false)
PrincipalName	A string containing the name of the principal (UPN) that requested the certificate. Typically, this field is only populated for end user certificates requested through Keyfactor Command (e.g. Mac auto-enrollment certificates).
CertRequestId	An integer containing the Keyfactor Command reference ID of the certificate request.
RequesterName	A string containing the name of the identity that requested the certificate.
ContentBytes	A string containing the certificate as bytes.
ExtendedKeyUsages	An array of objects containing the extended key usages associated with the certificate. Extended Key data includes:

Name	Description
Id	An integer containing the Keyfactor Command reference ID of the extended key usage.
Oid	A string indicating the OID of the extended key usage.
DisplayName	A string indicating the name of the extended key usage.

Name	Description
SubjectAltNameElements	An array of objects containing the subject alternative name elements of the certificate. SAN data includes:

Name	Description
Id	An integer containing the Keyfactor Command reference ID of the SAN Element.
Value	A string indicating the value of the SAN Element.
Type	An integer containing the type of SAN element. The possible values are:

Value	Description
0	Other Name
1	RFC 822 Name
2	DNS Name
3	X400 Address
4	Directory Name
5	Ediparty Name
6	Uniform Resource Identifier
7	IP Address
8	Registered Id
100	MS_NTPrincipalName
101	MS_NTDSReplication
999	Unknown

ValueHash	A string indicating a hash of the SAN value.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CRLDistributionPoints</td><td>An array of objects containing the distribution points for the certificate revocation lists the certificate could be in. CRL distribution point data includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer containing the Keyfactor Command reference ID of the CRL distribution point.</td></tr><tr><td>URL</td><td>A string indicating the URL of the CRL distribution point.</td></tr><tr><td>URLHash</td><td>A string indicating a hash of the URL.</td></tr></table></td></tr>
<tr><td>LocationsCount</td><td>An array of objects containing a count of how many certificates are in each location type. This returns a list of type and count combination. For example:<pre>"LocationsCount": [
   {
      "Type": "IIS",
      "Count": 2
   },
   {
      "Type": "F5-SL-REST",
      "Count": 1
   }
]</pre></td></tr>
<tr><td>SSLLocations</td><td>An array of objects containing the locations where the certificate is found using SSL discovery. SSL location data includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>StorePath</td><td>A string indicating the machine where the certificate was discovered.</td></tr><tr><td>AgentPool</td><td>A string indicating the GUID of the orchestrator pool that performed the SSL scan on the endpoint where the certificate was discovered.</td></tr><tr><td>IPAddress</td><td>A string indicating the IP address where the</td></tr></table></td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>certificate was discovered.</td></tr><tr><td>Port</td><td>An integer indicating the port on which the certificate was discovered.</td></tr><tr><td>NetworkName</td><td>A string indicating the name of the SSL network that performed the SSL scan (discovery or monitoring) on the endpoint where the certificate was discovered.</td></tr></table>

Name	Description
Locations	

Name	Description
StoreMachine	A string indicating the machine on which the certificate store is located.
StorePath	A string indicating the path on the machine where the certificate store is located. The format of this will vary depending on the type of store.
StoreType	An integer indicating the type of certificate store the certificate is located in. Possible values are:

Value	Description
0	Java Keystore
2	PEM File
3	F5 SSL Profiles
4	IIS Roots
5	NetScaler
6	IIS Personal
7	F5 Web Server
8	IIS Revoked
9	F5 Web Server REST
10	F5 SSL Profiles REST
11	F5 CA Bundles REST
100	Amazon Web Services
101	File Transfer Protocol
1xx	User-defined certificate stores will be given a type ID over 101.

Alias	A string indicating the alias of the certificate in the certificate store.
ChainLevel	An integer stating how many certificates are below this certificate in the certificate chain stored at the given location.

Name	Description
Metadata	An object containing the metadata fields populated for the certificate.
CertificateKeyId	An integer indicating the Keyfactor Command reference ID for the private key, if one exists, and public key of the certificate.
CARowIndex	An integer containing the CA's reference ID for certificate. **Note:** The *CARowIndex* has been replaced by *CARecordId*, but will remain for backward compatibility. It will only contain a non-zero value for certificates issued by Microsoft CAs. For Microsoft CA certificates, the *CARowIndex* will be equal to the *CARecordId* value parsed to an integer.
CARecordId	A string containing the CA's reference ID for certificate.
DetailedKeyUsage	An object containing details of the key usage configured for the certificate. Detailed key usage data includes:

Name	Description
CrlSign	A Boolean that indicates whether CRL signing is enabled for the certificate (true) or not (false).
DataEncipherment	A Boolean that indicates whether data encipherment ("Allow encryption of user data" in a Microsoft template) is enabled for the certificate (true) or not (false).
DecipherOnly	A Boolean that indicates whether the key of the certificate is intended for decipherment only (true) or not (false).
DigitalSignature	A Boolean that indicates whether digital signature is enabled for the certificate (true) or not (false).
EncipherOnly	A Boolean that indicates whether the key of the certificate is intended for encipherment only (true) or not (false).
KeyAgreement	A Boolean that indicates whether the certificate is configured for key agree-

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>ment.</td></tr>
<tr><td>KeyCertSign</td><td>A Boolean that indicates whether the certificate is configured for certificate signing.</td></tr>
<tr><td>KeyEncipherment</td><td>A Boolean that indicates whether the certificate is configured for key encipherment.</td></tr>
<tr><td>NonRepudiation</td><td>A Boolean that indicates whether the certificate is configured for non-repudiation.</td></tr>
<tr><td>HexCode</td><td>A string containing the hexadecimal code representing the total key usage. For example, a value of a0 would indicate digital signature with key encipherment.</td></tr>
</table>
</td></tr>
<tr><td>KeyRecoverable</td><td>A Boolean that indicates whether the certificate key is recoverable (true) or not (false).</td></tr>
<tr><td>Curve</td><td>A string indicating the OID of the elliptic curve algorithm configured for the certificate, for ECC templates. Well-known OIDs include:
1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1
1.3.132.0.34 = P-384/secp384r1
1.3.132.0.35 = P-521/secp521r1</td></tr>
<tr><td>CertStoreTypeShortNames</td><td>An array of comma-separated strings indicating the certificate stores types associated with each certificate.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.7 GET Certificates Metadata Compare

The GET /Certificates/Metadata/Compare method is used to compare the value of a metadata field in a certificate stored in Keyfactor Command with a provided value. This can be used to prevent

exposing sensitive data while still providing functionality. For example, with this method, a metadata attribute can be used along with the certificate itself as a second authentication factor to a third-party application. This method returns HTTP 200 OK on a success with a response of *true* if the compared values match or *false* if they do not.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 239: GET Certificates Metadata Compare Input Parameters

Name	In	Description
certificateId	Query	**Required**. An integer containing the Keyfactor Command reference ID of the certificate containing the metadata value to be compared.
metadataFieldName	Query	**Required**. A string containing the name of the metadata field whose value should be compared.
value	Query	**Required**. A string containing the value for comparison.
collectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.8 GET Certificates ID History

The GET /Certificates/{id}/History method is used to return details for the history of transactions for a certificate with the specified ID. History records are stored for a certificate for a variety of activities including initial import or enrollment, revocation, key recovery, additions or removals from certificate stores, renewals, and certificate discoveries in various certificate stores. For more information about certificate history records, see Certificate Details on page 19.This method returns HTTP 200 OK on a success with certificate history details in the message body.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 240: GET Certificates {id} History Input Parameters

Name	In	Description
id	Path	**Required**. An integer containing the Keyfactor Command reference ID of the certificate.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *OperationStart*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 241: GET Certificates {id} History Response Data

Name	Description
Id	An integer containing the Keyfactor Command reference ID of the certificate.
OperationStart	The date, in UTC, on which the operation begin.
OperationEnd	The date, in UTC, on which the operation completed.
Username	The name of the user who initiated the transaction that created the history record (e.g. enrolled for the certificate, revoked the certificate), in DOMAIN\\username format.
Comment	A string containing a comment that provides more information about the history record. For example (for a metadata field): `"AppOwnerEmailAddress has been updated from 'john.smith@keyexample.com' to 'martha.jones@keyexample.com'"`
Action	A string naming the action that was taken. For example: `Metadata Updated`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.9 DELETE Certificates

The DELETE /Certificates method is used to delete multiple certificates from the Keyfactor Command database in one request. The certificate IDs should be supplied in the request body as a JSON array of integers. If the specified certificate(s) have associated private key(s) stored in the database, these private keys are also removed. This endpoint returns 204 with no content upon success. IDs of any certificates that could not be deleted are returned in the response body. Delete operations will continue until the entire array of IDs has been processed.

Whenever a certificate is deleted that is a part of a certificate renewal chain. The certificates on either end of the deleted cert(s) will have their certificate histories updated to show that either a certificate before or after the certificate was deleted in the renewal chain of that certificate.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/delete/
OR

/certificates/collections/delete/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Tip: Deleting a certificate with this method does not necessarily delete it permanently. The certificate will be returned to the Keyfactor Command database on the next full synchronization if synchronization for the certificate source (certificate authority, SSL endpoint, etc.) is still configured. Certificate history, metadata, and private keys do not return when certificates re-synchronize. The certificate will be assigned a different Keyfactor Command reference ID when re-added to Keyfactor Command.

Table 242: DELETE Certificates Input Parameters

Name	In	Description
ids	Body	**Required**. An array of integers containing the Keyfactor Command certificate IDs for certificates that should be deleted in the form: `[123,789,567]` Use the *GET /Certificates* method (see GET Certificates below) to determine the certificate IDs.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.10 GET Certificates

The GET /Certificates method is used to return a list of certificates with certificate details. Results can be limited to selected keys using filtering, and URL parameters can be used to specify paging

and the level of information detail. This method returns HTTP 200 OK on a success with the requested certificates, as determined by filtering, and their certificate details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 243: GET Certificates Input Parameters

Name	In	Description
includeLocations	Query	A Boolean that sets whether to include the *Locations* data in the response (true) or not (false). If false is selected, the *LocationsCount* and *Locations* fields will still appear in the response, but they will contain no data. The default is *false*.
includeMetadata	Query	A Boolean that sets whether to include the *Metadata* data in the response (true) or not (false). If false is selected, the *Metadata* field will still appear in the response, but it will contain no data. The default is *false*.
includeHasPrivateKey	Query	A Boolean that sets whether to include the correct value for *HasPrivateKey* in the response (true) or not (false). If false is selected, the *HasPrivateKey* field will appear in the response with a value of *false* regardless of whether the certificate actually has a stored private key or not. The default is *false*.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.
includeRevoked	Query	A Boolean that sets whether to include revoked certificates in the results (true) or not (false). The default is *false*.
includeExpired	Query	A Boolean that sets whether to include expired certificates in the results (true) or not (false). The default is *false*.
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • ArchivedKey • CertId • CA • CertState

Name	In	Description
		• CertStoreContainer • CertStoreFQDN (alias: JavaKeystoreFQDN) • CertStorePath (alias: JavaKeystorePath) • CN (alias: IssuedCN) • DN (alias: IssuedDN) • ExpirationDate (alias: NotAfter) • EKU • EKUName • HasPrivateKey • ImportDate • IssuedDate (aliases: NotBefore and EffectiveDate) • IssuerDN • KeySize (alias: KeySizeInBits) • KeyType • KeyUsage • OU • NetBIOSPrincipal (alias: PrincipalName) • PublicKey • NetBIOSRequester (alias: RequesterName) • RevocationDate (alias: RevocationEffDate) • Revoker • RFC2818Compliant • SelfSigned • SerialNumber • SigningAlgorithm • SSLDNSName • SSLIPAddress (alias: SslHostName) • SSLNetworkName • SSLPort • SAN • TemplateDisplayName (alias: TemplateName) • TemplateShortName • Thumbprint The following fields have been deprecated and will be ignored if included in a request: • *CARequestID* • *CertRequestId*

Name	In	Description
		• *IsPfx* • *RequestResolutionDate* **Note:** Queries may be done using either the primary field name or the field alias(es).
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 244: GET Certificates Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the certificate.
Thumbprint	A string indicating the thumbprint of the certificate.
SerialNumber	A string indicating the serial number of the certificate.
IssuedDN	A string indicating the distinguished name of the certificate.
IssuedCN	A string indicating the common name of the certificate.
ImportDate	The date, in UTC, on which the certificate was imported into Keyfactor Command.
NotBefore	The date, in UTC, on which the certificate was issued by the certificate authority.
NotAfter	The date, in UTC, on which the certificate expires.
IssuerDN	A string indicating the distinguished name of the issuer.
PrincipalId	An integer indicating the Keyfactor Command reference ID of the principal (UPN) that requested the certificate. Typically, this field is only populated for end user certificates requested through Keyfactor Command (e.g. Mac auto-enrollment certificates). See also *PrincipalName*.
TemplateId	An integer indicating the Keyfactor Command reference ID of the template associated with the certificate.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CertState</td><td>An integer specifying the state of the certificate. The possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Active</td></tr><tr><td>2</td><td>Revoked</td></tr><tr><td>3</td><td>Denied</td></tr><tr><td>4</td><td>Failed</td></tr><tr><td>5</td><td>Pending</td></tr><tr><td>6</td><td>Certificate Authority</td></tr><tr><td>7</td><td>Parent Certificate Authority</td></tr></table></td></tr>
<tr><td>KeySizeInBits</td><td>An integer specifying the key size in bits.</td></tr>
<tr><td>KeyType</td><td>An integer specifying the key type of the certificate. The possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>RSA</td></tr><tr><td>2</td><td>DSA</td></tr><tr><td>3</td><td>ECC</td></tr><tr><td>4</td><td>DH</td></tr></table></td></tr>
<tr><td>RequesterId</td><td>An integer indicating the Keyfactor Command reference ID of the identity that requested the certificate. See also RequesterName.</td></tr>
<tr><td>IssuedOU</td><td>A string indicating the organizational unit of the certificate.</td></tr>
<tr><td>IssuedEmail</td><td>A string indicating the email address of the certificate.</td></tr>
<tr><td>KeyUsage</td><td>An integer indicating the total key usage of the certificate. Key usage is</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>stored in Active Directory as a single value made of a combination of values. The values that make up the key usage value include:
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></tr>
<tr><td>0</td><td>None</td><td>No key usage parameters.</td></tr>
<tr><td>1</td><td>Encipherment Only</td><td>The key can be used for encryption only.</td></tr>
<tr><td>2</td><td>CRL Signing</td><td>The key can be used to sign a certificate revocation list (CRL).</td></tr>
<tr><td>4</td><td>Key Certificate Signing</td><td>The key can be used to sign certificates.</td></tr>
<tr><td>8</td><td>Key Agreement</td><td>The key can be used to determine key agreement, such as a key created using the Diffie-Hellman key agreement algorithm.</td></tr>
<tr><td>16</td><td>Data Encipherment</td><td>The key can be used for data encryption.</td></tr>
<tr><td>32</td><td>Key Encipherment</td><td>The key can be used for key encryption.</td></tr>
<tr><td>64</td><td>Nonrepudiation</td><td>The key can be used for authentication.</td></tr>
<tr><td>128</td><td>Digital Signature</td><td>The key can be used as a digital signature.</td></tr>
<tr><td>32768</td><td>Decipherment Only</td><td>The key can be used for decryption only.</td></tr>
</table>
For example, a value of 160 would represent a key usage of digital signature with key encipherment. A value of 224 would add nonrepudiation to those.</td></tr>
<tr><td>SigningAlgorithm</td><td>A string indicating the algorithm used to sign the certificate.</td></tr>
<tr><td>CertStateString</td><td>A string containing the certificate state. The possible values are:
Unknown (0)</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

- Active (1)
- Revoked (2)
- Denied (3)
- Failed (4)
- Pending (5)
- Certificate Authority (6)
- Parent Certificate Authority (7)
- External Validation (8)

</td></tr>
<tr><td>KeyTypeString</td><td>A string containing the key type description (e.g. RSA) as per the types and descriptions shown for KeyType.</td></tr>
<tr><td>RevocationEffDate</td><td>The date, in UTC, on which the certificate was revoked, if applicable.</td></tr>
<tr><td>RevocationReason</td><td>An integer indicating the reason the certificate was revoked. The possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Unspecified</td></tr>
<tr><td>1</td><td>Key Compromised</td></tr>
<tr><td>2</td><td>CA Compromised</td></tr>
<tr><td>3</td><td>Affiliation Changed</td></tr>
<tr><td>4</td><td>Superseded</td></tr>
<tr><td>5</td><td>Cessation Of Operation</td></tr>
<tr><td>6</td><td>Certificate Hold</td></tr>
<tr><td>999</td><td>Unknown</td></tr>
</table>
</td></tr>
<tr><td>RevocationComment</td><td>An internally used Keyfactor Command field.</td></tr>
<tr><td>CertificateAuthorityId</td><td>An integer indicating the Keyfactor Command reference ID of the certificate authority that issued the certificate.</td></tr>
<tr><td>CertificateAuthorityName</td><td>A string indicating the certificate authority that issued the certificate.</td></tr>
<tr><td>TemplateName</td><td>A string indicating the display name of the template that was used when issuing the certificate.</td></tr>
</table>

Name	Description
ArchivedKey	A Boolean that indicates whether the certificate has a key archived in the issuing CA (true) or not (false).
HasPrivateKey	A Boolean that indicates whether the certificate has a private key stored in Keyfactor Command (true) or not (false)
PrincipalName	A string containing the name of the principal (UPN) that requested the certificate. Typically, this field is only populated for end user certificates requested through Keyfactor Command (e.g. Mac auto-enrollment certificates).
CertRequestId	An integer containing the Keyfactor Command reference ID of the certificate request.
RequesterName	A string containing the name of the identity that requested the certificate.
ContentBytes	A string containing the certificate as bytes.
ExtendedKeyUsages	An array of objects containing the extended key usages associated with the certificate. Extended Key data includes:

Name	Description
Id	An integer containing the Keyfactor Command reference ID of the extended key usage.
Oid	A string indicating the OID of the extended key usage.
DisplayName	A string indicating the name of the extended key usage.

Name	Description
SubjectAltNameElements	An array of objects containing the subject alternative name elements of the certificate. SAN data includes:

Name	Description
Id	An integer containing the Keyfactor Command reference ID of the SAN Element.
Value	A string indicating the value of the SAN Element.
Type	An integer containing the type of SAN element. The possible values are:

Value	Description
0	Other Name
1	RFC 822 Name
2	DNS Name
3	X400 Address
4	Directory Name
5	Ediparty Name
6	Uniform Resource Identifier
7	IP Address
8	Registered Id
100	MS_NTPrincipalName
101	MS_NTDSReplication
999	Unknown

ValueHash	A string indicating a hash of the SAN value.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CRLDistributionPoints</td><td>An array of objects containing the distribution points for the certificate revocation lists the certificate could be in. CRL distribution point data includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer containing the Keyfactor Command reference ID of the CRL distribution point.</td></tr><tr><td>URL</td><td>A string indicating the URL of the CRL distribution point.</td></tr><tr><td>URLHash</td><td>A string indicating a hash of the URL.</td></tr></table></td></tr>
<tr><td>LocationsCount</td><td>An array of objects containing a count of how many certificates are in each location type. This returns a list of type and count combination. For example:<pre>"LocationsCount": [
   {
      "Type": "IIS",
      "Count": 2
   },
   {
      "Type": "F5-SL-REST",
      "Count": 1
   }
]</pre></td></tr>
<tr><td>SSLLocations</td><td>An array of objects containing the locations where the certificate is found using SSL discovery. SSL location data includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>StorePath</td><td>A string indicating the machine where the certificate was discovered.</td></tr><tr><td>AgentPool</td><td>A string indicating the GUID of the orchestrator pool that performed the SSL scan on the endpoint where the certificate was discovered.</td></tr><tr><td>IPAddress</td><td>A string indicating the IP address where the</td></tr></table></td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>certificate was discovered.</td></tr><tr><td>Port</td><td>An integer indicating the port on which the certificate was discovered.</td></tr><tr><td>NetworkName</td><td>A string indicating the name of the SSL network that performed the SSL scan (discovery or monitoring) on the endpoint where the certificate was discovered.</td></tr></table>

Name	Description
Locations	

Name	Description
StoreMachine	A string indicating the machine on which the certificate store is located.
StorePath	A string indicating the path on the machine where the certificate store is located. The format of this will vary depending on the type of store.
StoreType	An integer indicating the type of certificate store the certificate is located in. Possible values are:

Value	Description
0	Java Keystore
2	PEM File
3	F5 SSL Profiles
4	IIS Roots
5	NetScaler
6	IIS Personal
7	F5 Web Server
8	IIS Revoked
9	F5 Web Server REST
10	F5 SSL Profiles REST
11	F5 CA Bundles REST
100	Amazon Web Services
101	File Transfer Protocol
1xx	User-defined certificate stores will be given a type ID over 101.

Alias	A string indicating the alias of the certificate in the certificate store.
ChainLevel	An integer stating how many certificates are below this certificate in the certificate chain stored at the given location.

Name	Description
Metadata	An object containing the metadata fields populated for the certificate.
CertificateKeyId	An integer indicating the Keyfactor Command reference ID for the private key, if one exists, and public key of the certificate.
CARowIndex	An integer containing the CA's reference ID for certificate. **Note:** The *CARowIndex* has been replaced by *CARecordId*, but will remain for backward compatibility. It will only contain a non-zero value for certificates issued by Microsoft CAs. For Microsoft CA certificates, the *CARowIndex* will be equal to the *CARecordId* value parsed to an integer.
CARecordId	A string containing the CA's reference ID for certificate.
DetailedKeyUsage	An object containing details of the key usage configured for the certificate. Detailed key usage data includes:

Name	Description
CrlSign	A Boolean that indicates whether CRL signing is enabled for the certificate (true) or not (false).
DataEncipherment	A Boolean that indicates whether data encipherment ("Allow encryption of user data" in a Microsoft template) is enabled for the certificate (true) or not (false).
DecipherOnly	A Boolean that indicates whether the key of the certificate is intended for decipherment only (true) or not (false).
DigitalSignature	A Boolean that indicates whether digital signature is enabled for the certificate (true) or not (false).
EncipherOnly	A Boolean that indicates whether the key of the certificate is intended for encipherment only (true) or not (false).
KeyAgreement	A Boolean that indicates whether the certificate is configured for key agree-

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>ment.</td></tr>
<tr><td>KeyCertSign</td><td>A Boolean that indicates whether the certificate is configured for certificate signing.</td></tr>
<tr><td>KeyEncipherment</td><td>A Boolean that indicates whether the certificate is configured for key encipherment.</td></tr>
<tr><td>NonRepudiation</td><td>A Boolean that indicates whether the certificate is configured for non-repudiation.</td></tr>
<tr><td>HexCode</td><td>A string containing the hexadecimal code representing the total key usage. For example, a value of a0 would indicate digital signature with key encipherment.</td></tr>
</table>
</td></tr>
<tr><td>KeyRecoverable</td><td>A Boolean that indicates whether the certificate key is recoverable (true) or not (false).</td></tr>
<tr><td>Curve</td><td>A string indicating the OID of the elliptic curve algorithm configured for the certificate, for ECC templates. Well-known OIDs include:
1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1
1.3.132.0.34 = P-384/secp384r1
1.3.132.0.35 = P-521/secp521r1</td></tr>
<tr><td>CertStoreTypeShortNames</td><td>An array of comma-separated strings indicating the certificate stores types associated with each certificate.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.11 PUT Certificates Metadata

The PUT /Certificates/Metadata method is used to update one or more metadata values for a specified certificate. Any existing values for the metadata fields submitted with this update will be

overwritten with the new values provided. For more granular control over updating only metadata fields that do not already contain a value, use the *PUT /Certificates/Metadata/All* method (see PUT Certificates Metadata All on the next page). This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/metadata/modify/
OR
/certificates/collections/metadata/modify/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 245: PUT Certificates Metadata Input Parameters

Name	In	Description
Id	Body	**Required**. An integer specifying the Keyfactor Command reference ID for the certificate to update.

Metadata	Body	**Required**. An object containing one or more metadata key value pairs to update for the certificate. These are submitted with the metadata field name in the key and the value in the value. For example:
		`"Metadata": {`
		`"AppOwnerEmailAddress":"john.smith@keyexample.com", // String field`
		`"SiteCode":23, // Integer fie`
		`"BusinessCritical":true, // Boolean fie`
		`"Notes":"Here are some notes about this certificate.", // BigText field`
		`"BusinessUnit":"E-Business", // Multiple Choice fiel pre-defined value`
		`"TicketResolutionDate":"2021-07-23" // Date field in yyyy-mm-dd format`
		`}`

CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.12 PUT Certificates Metadata All

The PUT /Certificates/Metadata/All method is used to update one or more metadata values for a specified set of certificates. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/metadata/modify/
OR
/certificates/collections/metadata/modify/#/ (where # is a reference to a specific certificate collection ID)

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 246: PUT Certificates Metadata All Input Parameters

Name	In	Description
Query	Body	**Required***. A string containing a query to limit the certificates to update (e.g. field1 -eq value1 AND field2 -gt value2). Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns. For querying guidelines, refer to: Certificate Search Page on page 34. A value for one of *CertificateIds*, *Query*, or *CollectionId* is **required**. The query fields supported for this endpoint are: • ArchivedKey • CertId • CA • CertState • CertStoreContainer • CertStoreFQDN (alias: JavaKeystoreFQDN) • CertStorePath (alias: JavaKeystorePath) • CN (alias: IssuedCN) • DN (alias: IssuedDN) • ExpirationDate (alias: NotAfter)

Name	In	Description
		• EKU • EKUName • HasPrivateKey • ImportDate • IssuedDate (aliases: NotBefore and EffectiveDate) • IssuerDN • KeySize (alias: KeySizeInBits) • KeyType • KeyUsage • OU • NetBIOSPrincipal (alias: PrincipalName) • PublicKey • NetBIOSRequester (alias: RequesterName) • RevocationDate (alias: RevocationEffDate) • Revoker • RFC2818Compliant • SelfSigned • SerialNumber • SigningAlgorithm • SSLDNSName • SSLIPAddress (alias: SslHostName) • SSLNetworkName • SSLPort • SAN • TemplateDisplayName (alias: TemplateName) • TemplateShortName • Thumbprint The following fields have been deprecated and will be ignored if included in a request: • *CARequestID* • *CertRequestId* • *IsPfx* • *RequestResolutionDate* **Note:** Queries may be done using either the primary field name or the field alias(es).

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>Tip: To exclude revoked certificates from the update, include a query of:

<code>CertState -ne \"2\"</code>

To exclude expired certificates from the update, include a query of:

<code>ExpirationDate -ge \"%TODAY%\"</code></td></tr>
<tr><td>CertificateIds</td><td>Body</td><td>Required*. An array of integers indicating the Keyfactor Command certificate IDs to update. A value for one of CertificateIds, Query, or CollectionId is required.</td></tr>
<tr><td>Metadata</td><td>Body</td><td>Required. An array of objects containing information about the metadata field(s) to update. The parameters are:

<table><tr><th>Name</th><th>Description</th></tr><tr><td>Value</td><td>Required. A string indicating the value that should be set for the metadata field.</td></tr><tr><td>MetadataName</td><td>Required. A string indicating the name of the metadata field that should be updated for the certificates.</td></tr><tr><td>OverwriteExisting</td><td>A Boolean that sets whether all the certificates being updated will have their metadata field overwritten to the value being provided, including those that already have a value in the given metadata field (true) or whether only the certificates that currently have no value in the given metadata field will be saved with the new value (false). The default is false.</td></tr></table>
For example:

<pre>"Metadata": [
  {</pre></td></tr>
</table>

Name	In	Description

		"MetadataName": "AppOwnerEmailAddress", // This is a String field.
		"Value": "john.smith@keyexample.com",
		"OverwriteExisting": true
		},
		{
		"MetadataName": "SiteCode", // This is an Integer field.
		"Value": 5,
		"OverwriteExisting": true
		},
		{
		"MetadataName": "BusinessCritical", // This is a Boolean field.
		"Value": true,
		"OverwriteExisting": true
		},
		{
		"MetadataName": "Notes", // This is a BigText field.
		"Value": "Here are some notes about this certificate.",
		"OverwriteExisting": true
		},
		{
		"MetadataName": "BusinessUnit", // This is a Multiple Choice field.
		"Value": "E-Business", // This is a value pre-defined for the field.
		"OverwriteExisting": true
		},
		{
		"MetadataName": "TicketResolutionDate", // This is a Date field in yyyy-mm-dd format.
		"Value": "2021-07-23",
		"OverwriteExisting": true
		}
		]

CollectionId	Query	**Required***. An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information. This field can also be used to specify the certificate collection containing certificates that should be updated. A value for one of *CertificateIds*, *Query*, or *CollectionId* is **required**.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.13 POST Certificates Import

The POST /Certificates/Import method is used to import a certificate provided in the request body into Keyfactor Command. This method returns HTTP 200 OK on a success with a message body containing information about the import.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/import/

Table 247: POST Certificates Import Input Parameters

Name	In	Description
Certificate	Body	**Required**. A string containing the base-64 encoded contents of the certificate that is to be imported into Keyfactor Command.
Password	Body	**Required***. A string containing the password used to decrypt the imported PFX. This field is **required** if a PFX certificate is provided in the *Certificate* field.

Metadata	Body	An object containing the certificate metadata that will be associated with the certificate once it is imported. This is provided as a set of key value pairs with the metadata field name in the key and the value in the value. For example:
		`"Metadata": {`
		`"AppOwnerFirstName": "John",`
		`"AppOwnerLastName": "Smith"`
		`}`

StoreIds	Body	An array of strings indicating the certificate store GUIDs that the imported certificate will be installed into.

Name	In	Description
StoreTypes	Body	

Name	Description
StoreTypeId	An integer indicating the ID of the store type being used. There must be one for each type of store ID used. The possible values are:

Value	Description
0	Java Keystore
2	PEM File
3	F5 SSL Profiles
4	IIS Roots
5	NetScaler
6	IIS Personal
7	F5 Web Server
8	IIS Revoked
9	F5 Web Server REST
10	F5 SSL Profiles REST
11	F5 CA Bundles REST
100	Amazon Web Services
101	File Transfer Protocol
1xx	User-defined certificate stores will be given a type ID over 101.

Alias	**Required***. A string providing an alias to be used for the certificate upon entry into the certificate store. The function of the alias varies depending on the certificate store type. For example, for an F5 device, it serves as the file name used to store the file in the device file system, minus the extension (e.g. use alias MyFile for a file named MyFile.pfx) while for a Java keystore, it is stored in the keystore associated with the certificate. Some certificate store types don't require an alias and some do. See Add Certificate on page 74 for more information. This field may be **required** depending on the store type selected.
Overwrite	A Boolean that sets whether a certificate in the store with the *Alias* provided should be overwritten with the certificate being imported (true) or not (false).

Name	In	Description
Schedule	Body	A string containing the time the imported certificate should be scheduled to be installed into the certificate store. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

Table 248: POST Certificates Import Response Data

Name	Description
ImportStatus	An integer indicating the status of the import job indicating, for example, whether the certificate was newly created in Keyfactor Command or already existed in Keyfactor Command and was just updated based on provided private key, metadata, or location information.
InvaildKeyStores	An array of objects indicating which key store items failed with some information. Included parameters are: **Name** / **Description** KeystoreId — A string indicating the ID of the store that failed. ClientMachine — A string indicating the client machine of the store that failed. StorePath — A string indicating the path to the location of the certificate store that failed. Alias — A string indicating the alias for the certificate in the store that failed. Reason — An integer indicating the simple reason why it failed. Explanation — A string indicating a more specific reason for the failure.
JobStatus	An integer indicating the state of all certificate store jobs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.14 POST Certificates Revoke

The POST /Certificates/Revoke method is used to revoke one or more certificates with the specified ID(s). This method returns HTTP 200 OK on a success with a list of the successfully revoked certificate IDs on a success or a list of the failed certificate IDs if any revocations fail.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/revoke/
OR
/certificates/collections/revoke/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

Table 249: POST Certificates Revoke Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>CertificateIDs</td><td>Body</td><td>Required. An array of integers containing the list of Keyfactor Command reference IDs for certificates that should be revoked.</td></tr>
<tr><td>Reason</td><td>Body</td><td>An integer containing the specific reason that the certificate is being revoked. Available values are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>-1</td><td>Remove from Hold</td></tr><tr><td>0</td><td>Unspecified</td></tr><tr><td>1</td><td>Key Compromised</td></tr><tr><td>2</td><td>CA Compromised</td></tr><tr><td>3</td><td>Affiliation Changed</td></tr><tr><td>4</td><td>Superseded</td></tr><tr><td>5</td><td>Cessation of Operation</td></tr><tr><td>6</td><td>Certificate Hold</td></tr><tr><td>7</td><td>Remove from CRL. Only valid in the case that a cert is already on a CRL in a manner that it can be removed, such as Certificate Hold</td></tr></table>
The default is Unspecified.</td></tr>
<tr><td>Comment</td><td>Body</td><td>Required. A string containing a freeform reason or comment on why the certificate is being revoked.</td></tr>
<tr><td>EffectiveDate</td><td>Body</td><td>A string containing the date and time when the certificate will be revoked. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). The default is the current date and time.</td></tr>
<tr><td>CollectionId</td><td>Body</td><td>An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See <a>Certificate Collection Permissions on page 627 for more information.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.15 POST Certificates Analyze

The POST /Certificates/Analyze method is used to parse a raw binary certificate returned from a CA into human-readable list of certificate details. For input data supplied with chain certificates, the output will include analysis of the primary certificate and the chain certificates. This method returns HTTP 200 OK on a success with a list of the contents of the certificate and the certificates in the chain, if applicable.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
OR
/certificates/import/
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 250: POST Certificates Analyze Input Parameters

Name	In	Description
Certificate	Body	**Required**. A string containing either the PEM-encoded string of the certificate **not** including the header and footer (e.g. -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----) or the base-64 encoded contents of the certificate. Both certificates with and without private keys are supported for analysis.
Password	Body	A string containing the password used to encrypt the private key of the certificate, if applicable.

Table 251: POST Certificates Analyze Response Data

Name	Description
IssuedDN	A string containing the distinguished name of the certificate.
IssuerDN	A string containing the distinguished name of the issuer of the certificate.
Thumbprint	A string containing the thumbprint of the certificate.
NotAfter	A string containing the date/time, in UTC, on which the certificate expires.
NotBefore	A string containing the date/time, in UTC, on which the certificate was issued by the certificate authority.
Metadata	An object containing the metadata fields populated for the certificate.
IsEndEntity	A Boolean indicating whether the certificate is the end entity of the chain (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.16 POST Certificates Recover

The POST /Certificates/Recover method is used to recover or download a certificate with private key. For certificates that are available for key recovery from the Microsoft CA, the certificate is recovered from the CA. For certificates with a private key stored in Keyfactor Command, the certificate is downloaded from Keyfactor Command. This method returns HTTP 200 OK on a success with a base-64-encoded representation of the certificate and private key, including optional certificate chain, in JKS, PEM or PFX format. For certificates without private keys in DER, PEM or P7B format, use the *POST /Certificates/Download* method (see POST Certificates Download on page 1173).

Tip: CA-level key recovery is supported for Microsoft CAs to allow recovery of private keys for certificates enrolled outside of Keyfactor Command. CA-level key archiving is not supported for enrollments done through Keyfactor Command. CA-level key recovery is not supported for EJBCA CAs. For enrollments done through Keyfactor Command for either Microsoft or EJBCA CAs, use Keyfactor Command private key retention (see Details Tab on page 387).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/certificates/collections/private_key/read/
OR
/certificates/collections/private_key/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 252: POST Certificates Recover Input Parameters

Name	In	Description
Password	Body	**Required**. The password to set on the certificate.
CertID	Body	**Required***. An integer indicating the Keyfactor Command reference ID of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
SerialNumber	Body	**Required***. A string indicating the serial number of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
IssuerDN	Body	**Required***. A string indicating the distinguished name of the issuer of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
Thumbprint	Body	**Required***. A string indicating the thumbprint of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
IncludeChain	Body	A Boolean indicating whether to include the certificate chain with the certificate (true) or not (false). If you select *true*, you must select a certificate format of PEM or P7B.
ChainOrder	Body	A string indicating the order in which the certificate chain should be returned if *IncludeChain* is set to *true*. Supported values are *EndEntityFirst* or *RootFirst*.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user

Name	In	Description
		must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.
x-certificateformat	Header	The desired output format for the certificate. Supported options are: • JKS Selecting the JKS option allows you to create a Java keystore with the returned PFX value. • PEM Output the certificate in base-64 encoded PEM format along with the private key and any optional chain certificates in a single file. • PFX Selecting the PFX option allows you to create a PKCS#12 (PFX/P12) file with the returned PFX value.

Table 253: POST Certificates Recover Response Data

Name	Description

PFX	The base-64-encoded representation of the certificate in PEM or PFX format with the optional certificate chain. The string will need to be base-64 decoded for both PEM and PFX. This can be accomplished in a number of ways. For example, using PowerShell and a manually generated file containing just the base-64 string returned in the response (not the full response):
	`$b64 = Get-Content 'C:\path\to\source\file'`
	`$targetFile = 'C:\path\to\target\file'`
	
	`$bytes = [Convert]::FromBase64String($b64)`
	`[IO.File]::WriteAllBytes($targetFile, $bytes)`
	
	Using PowerShell within the script where the full response (including two key/value pairs) is returned and placed in the variable $response:
	`$ResponseContent = $response.Content | ConvertFrom-Json`
	
	`$targetFile = 'C:\path_to_target_file\'+$ResponseContent.FileName`
	
	`$bytes = [Convert]::FromBase64String($ResponseContent.PFX)`
	`[IO.File]::WriteAllBytes($targetFile, $bytes)`
	
	In the second case, the name provided in FileName is used for the PFX output file.

FileName	The CN of the certificate presented as a file name (e.g. mycertificatekeyexamplecom.pfx).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.17 POST Certificates Download

The POST /Certificates/Download method is used to download a certificate from Keyfactor Command. This method returns HTTP 200 OK on a success with the base-64-encoded certificate without private key, including optional certificate chain, in DER, PEM of P7B format. For certificates with private keys in PEM or PFX format, use the *POST /Certificates/Recover* method (see POST Certificates Recover on page 1169).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 254: POST Certificates Download Input Parameters

Name	In	Description
CertID	Body	**Required***. An integer indicating the Keyfactor Command reference ID of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
SerialNumber	Body	**Required***. A string indicating the serial number of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
IssuerDN	Body	**Required***. A string indicating the distinguished name of the issuer of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
Thumbprint	Body	**Required***. A string indicating the thumbprint of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
IncludeChain	Body	A Boolean indicating whether to include the certificate chain with the certificate (true) or not (false). If you select *true*, you must select a certificate format of PEM or P7B.
ChainOrder	Body	A string indicating the order in which the certificate chain should be returned if *IncludeChain* is set to *true*. Supported values are *EndEntityFirst* or *RootFirst*.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate

Name	In	Description
		collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.
x-certificateformat	Header	The desired output format for the certificate. Supported options are: • DER DER is not supported if IncludeChain is set to *true*. • PEM Output the certificate in base-64 encoded PEM format along with any optional chain certificates in a single file. • P7B This option is only supported if IncludeChain is set to *true*

Table 255: POST Certificates Download Response Data

Name	Description
Content	The base-64-encoded certificate in DER, PEM or P7B format with the optional certificate chain.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.18 POST Certificates Revoke All

The POST /Certificates/RevokeAll method is used to revoke all the certificates in the specified query and/or collection ID. The endpoint makes use of the *Revoke All Enabled* application setting (see Application Settings: Console Tab on page 602). If *Revoke All Enabled* is set to *False*, the endpoint will return an error indicating revoke all is not allowed and not complete the request. This method returns HTTP 200 OK on a success with a list of the successfully revoked certificate IDs on a success or a list of the failed certificate IDs if any revocations fail.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/revoke/
OR

/certificates/collections/revoke/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

Table 256: POST Certificates Revoke All Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Query</td><td>Body</td><td>Required*. A string containing a query to limit the certificates to revoke (e.g. field1 -eq value1 AND field2 -gt value2). Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns. For querying guidelines, refer to Certificate Search Page on page 34. A value for either Query or CollectionId is required. If both Query and CollectionId are specified, certificates from both sources will be revoked.</td></tr>
<tr><td>Reason</td><td>Body</td><td>An integer containing the specific reason that the certificates are being revoked. Available values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>-1</td><td>Remove from Hold</td></tr><tr><td>0</td><td>Unspecified</td></tr><tr><td>1</td><td>Key Compromised</td></tr><tr><td>2</td><td>CA Compromised</td></tr><tr><td>3</td><td>Affiliation Changed</td></tr><tr><td>4</td><td>Superseded</td></tr><tr><td>5</td><td>Cessation of Operation</td></tr><tr><td>6</td><td>Certificate Hold</td></tr><tr><td>7</td><td>Remove from CRL</td></tr></table>The default is Unspecified.</td></tr>
<tr><td>Comment</td><td>Body</td><td>Required. A string containing a freeform reason or comment indicating why the certificates are being revoked.</td></tr>
<tr><td>EffectiveDate</td><td>Body</td><td>The date and time when the certificate will be revoked. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). The default is the current date and time.</td></tr>
<tr><td>IncludeRevoked</td><td>Body</td><td>A Boolean that indicates whether revoked certificates should be included in the revocation (true) or not (false). The default is false.</td></tr>
<tr><td>IncludeExpired</td><td>Body</td><td>A Boolean that indicates whether expired certificates should be included in</td></tr>
</table>

Name	In	Description
		the revocation (true) or not (false). The default is *false*.
CollectionId	Query	**Required***. An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information. This field can also be used to specify the certificate collection containing certificates that should be revoked. A value for either *Query* or *CollectionId* is **required**. If both *Query* and *CollectionId* are specified, certificates from both sources will be revoked. For example: `https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/RevokeAll?CollectionId=14`

Note: This endpoint is not exposed in the Keyfactor API Reference and Utility to reduce accidental usage. You may still make use of it by calling it from your own tool.

3.6.7.19 DELETE Certificates Query

The DELETE /Certificates/query method is used to delete a group of certificates from Keyfactor Command that match the criteria provided in the body. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/delete/
OR
/certificates/collections/delete/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 257: DELETE Certificates Query Input Parameters

Name	In	Description
sq	Body	**Required**. Query to limit the requested set of certificates that should be deleted in the form (without parameter name): `CN -contains "mycertificate.keyexample.com"` See Certificate Search Page on page 34 for querying guidelines to build your body query.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.20 DELETE Certificates Private Key

The DELETE /Certificates/PrivateKey method is used to delete the stored private key of each certificate ID in the list provided in the body from the Keyfactor Command platform. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/delete/
OR
/certificates/collections/delete/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 258: DELETE Certificates Private Key Input Parameters

Name	In	Description
ids	Body	**Required**. An array of integers containing the Keyfactor Command reference IDs for certificates for which the associated private keys should be deleted in the form: `[123,789,567]` Use the *GET /Certificates* method (see GET Certificates on page 1141) to determine the certificate IDs.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.7.21 DELETE Certificates Private Key ID

The DELETE /Certificates/PrivateKey/{id} method is used to delete the stored private key of the submitted certificate ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/delete/
OR
/certificates/collections/delete/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 259: DELETE Certificates Private Key {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the certificate whose private key should be deleted. Use the *GET /Certificates* method (see GET Certificates on page 1141) to retrieve a list of certificates based on entered search criteria to determine the certificate ID. This information is also available in the certificate details for a certificate in the Keyfactor Command Management Portal.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8 Certificate Authority

The CertificateAuthority component of the Keyfactor API includes methods for listing, creating, updating and deleting certificate authority records in Keyfactor Command as well as for publishing CRLs.

Table 260: Certificate Authority Endpoints

Endpoint	Method	Description	Link
/PublishCRL	POST	Publishes the Certificate Revocation List of the given certificate authority.	POST Certificate Authority PublishCRL on page 1284
/	GET	Returns a list of all certificate author-	GET Certificate Authority on

Endpoint	Method	Description	Link
		ities.	page 1201
/	POST	Creates a new certificate authority record.	POST Certificate Authority on page 1216
/	PUT	Updates an existing certificate authority record.	PUT Certificate Authority on page 1248
/{id}	GET	Returns details for the certificate authority identified by the specified ID.	GET Certificate Authority ID on page 1186
/{id}	DELETE	Deletes the certificate authority record for the specified ID.	DELETE Certificate Authority ID on page 1185
/Test	POST	Validates that the certificate authority with the provided information can be reached.	POST Certificate Authority Test on page 1280
/SourceCount	GET	Retrieve the count of certificate authorities with full or incremental synchronization scans configured.	GET Certificate Authority Source Count on page 1284
/AvailableForests	GET	Retrieve the list of forests in Active Directory Keyfactor Command	GET Certificate Authority Available Forests on page 1285
/Import	POST	Import into Keyfactor Command any certificate authorities from the provided configuration tenant DNS suffix	POST Certificate Authority Import on page 1294

Endpoint	Method	Description	Link
		(e.g. keyexample.com).	
/HealthMonitoring/Schedule	GET	Retrieve the current schedule for the CA health monitoring job.	GET Certificate Authority Health Monitoring Schedule on page 1286
/AlertRecipients/CAHealthRecipients	POST	Create new recipients to receive CA health monitoring alerts in Keyfactor Command	POST Certificate Authority Alert Recipients CA Health Recipients on page 1287
/AlertRecipients/CAHealthRecipients	GET	Retrieve the list of recipients configured in Keyfactor Command for CA health monitoring alerts.	GET Certificate Authority Alert Recipients CA Health Recipients on page 1286
/AlertRecipients/CAHealthRecipients/{id}	GET	Retrieve the CA health monitoring recipient configured in Keyfactor Command with the specified ID.	GET Certificate Authority Alert Recipients CA Health Recipients ID on page 1288
/AlertRecipients/CAHealthRecipients/{id}	PUT	Update the CA health monitoring alert recipient with the specified ID.	PUT Certificate Authority Alert Recipients CA Health Recipients ID on page 1290
/AlertRecipients/CAHealthRecipients/{id}	DELETE	Delete the CA threshold recipient with the specified Keyfactor Command reference ID.	DELETE Certificate Authority Alert Recipients CA Health Recipients ID on page 1289
/AlertRecipients/CAThresholdRecipients	POST	Create new recipients to receive CA	POST Certificate Authority Alert Recip-

Endpoint	Method	Description	Link
		threshold alerts in Keyfactor Command.	ients CA Threshold Recipients on page 1292
/AlertRecipients/CAThresholdRecipients	GET	Retrieve the list of recipients configured in Keyfactor Command for CA threshold alerts.	GET Certificate Authority Alert Recipients CA Threshold Recipients on page 1291
/AlertRecipients/CAThresholdRecipients/{id}	GET	Retrieve the CA threshold recipient configured in Keyfactor Command with the specified ID.	GET Certificate Authority Alert Recipients CA Threshold Recipients ID on page 1292
/AlertRecipients/CAThresholdRecipients/{id}	PUT	Update the CA threshold alert recipient with the specified ID.	PUT Certificate Authority Alert Recipients CA Threshold Recipients ID on page 1293
/AlertRecipients/CAThresholdRecipients/{id}	DELETE	Delete the CA threshold recipient with the specified Keyfactor Command reference ID.	DELETE Certificate Authority Alert Recipients CA Threshold Recipients ID on page 1290

3.6.8.1 DELETE Certificate Authority ID

The DELETE /CertificateAuthority/{id} endpoint is used to delete the certificate authority record with the specified Keyfactor Command reference ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Note: A CA that has active records associated with it (e.g. certificates, certificate requests) cannot be deleted from Keyfactor Command.

Table 261: DELETE Certificate Authority {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer that specifies the Keyfactor Command reference ID of the certificate authority record to delete.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.2 GET Certificate Authority ID

The POST /CertificateAuthority method is used to retrieve details for a specified certificate authority. This method returns HTTP 200 OK on a success with the details for the certificate authority.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 262: GET Certificate Authority {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer that specifies the Keyfactor Command ID of the certificate authority record to retrieve.

Table 263: GET Certificate Authority {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command identifier for the certificate authority. The ID is automatically assigned by Keyfactor Command.
LogicalName	A string indicating the logical name of the certificate authority.
HostName	A string indicating the DNS hostname (for DCOM configurations) or URL (for HTTPS configurations) of the certificate authority (e.g. myca.keyexample.com or https://myca.keyexample.com).
Delegate	A Boolean that sets whether management interactions with the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, these interactions are done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
DelegateEnrollment	A Boolean that sets whether enrollment to the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, enrollment is done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
ForestRoot	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). **Note:** This field is retained for legacy purposes and will auto-populate with the value provided in the *ConfigurationTenant* field.
ConfigurationTenant	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). For EJBCA CAs, this is a reference ID and does not need to be the DNS domain name. The short hostname of the EJBCA CA server makes a good reference ID.

Name	Description
	Important: EJBCA and Microsoft CAs cannot be configured with the same *Configuration Tenant*, so do not set this to the DNS domain name for an EJBCA CA if you will also be configuring Microsoft CAs in the same DNS domain.
Remote	A Boolean that sets whether communications with the certificate authority are done via a Keyfactor Universal Orchestrator configured to manage remote CAs. If set to *true*, a value must be provided for the *Agent*. The default is *false*.
Agent	A string indicating the GUID of the Keyfactor Universal Orchestrator configured to manage the certificate authority (see *Remote*).
Standalone	A Boolean that sets whether the certificate authority is a standalone CA (*true*) or not (*false*). If both *Standalone* is set to *true* and *AllowedEnrollmentTypes* is set to 1 or 3, *KeyRetention* may be set. The default is *false*.
MonitorThresholds	A Boolean that sets whether threshold monitoring is enabled. If set to *true*, email alerts will be sent when certificate issuance or failures (including denials) since the last threshold alert was sent falls outside the configured limits. If this option is set to *true*, the following additional fields should also be set: • IssuanceMax • IssuanceMin • FailureMax The DenialMax field has been deprecated and should always be zero. Monitoring is not supported for CAs accessed with the Keyfactor Universal Orchestrator. The default is *false*. See also *ThresholdCheck* to configure the monitoring frequency. **Note:** For full functionality of threshold monitoring, you must also configure email recipients for threshold alerts. These are configured globally rather than on a CA-by-CA basis. See Certificate Authority Monitoring on page 378 for more information.
IssuanceMax	An integer that sets the maximum number of certificates that can be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.

Name	Description
IssuanceMin	An integer that sets the minimum number of certificates that should be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If fewer certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
DenialMax	This is considered deprecated and may be removed in a future release.
FailureMax	An integer that sets the maximum number of certificate requests that can fail or be denied in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificate requests than this fail in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
RFCEnforcement	A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (*true*) or not (*false*). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. The default is *false*. **Tip:** The *RFCEnforcement* option at the CA level is used only for standalone CAs. RFC enforcement for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435).
Properties	A string indicating additional properties about the certificate authority. This field is used to store the configuration for the *Sync External Certificates* option. This option allows foreign certificates that have been imported into a Microsoft CA to be synchronized to Keyfactor Command along with the certificates issued by the Microsoft CA. The setting is referenced using the following format: `{\"syncExternal\":true} OR {\"syncExternal\":false}`
AllowedEnrollmentTypes	An integer that sets the type(s) of enrollment that are allowed through Keyfactor Command for the certificate authority. Possible values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>PFX and CSR Enrollment</td></tr>
</table>
This value is unset by default.
</td></tr>
<tr><td>KeyRetention</td><td>
An integer that sets the type of key retention to enable for the certificate authority, if any. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Key Retention Disabled</td></tr>
<tr><td>1</td><td>Indefinite</td></tr>
<tr><td>2</td><td>After Expiration</td></tr>
<tr><td>3</td><td>From Issuance</td></tr>
</table>
Values of 2 and 3 require setting KeyRetentionDays.
This value is unset by default.
Tip: The KeyRetention option at the CA level is used only for standalone CAs. Key retention for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435). KeyRetention on a CA may only be set to a value other than zero if both Standalone is set to true and AllowedEnrollmentTypes is set to 1 or 3. Some level of private key retention must be configured when using PFX enrollment with a standalone CA. See Adding or Modifying a CA Record on page 354 for more information.
</td></tr>
<tr><td>KeyRetentionDays</td><td>An integer indicating the number of days for which to retain the private keys for certificates issued by this certificate authority before scheduling them for deletion. This value is unset by default.</td></tr>
<tr><td>ExplicitCredentials</td><td>A Boolean that sets whether explicit credentials are enabled for this certificate authority (true) or not (false). Set this to true for CAs that do not support integrated authentication or are not configured for integrated authentication and enter credentials in the ExplicitUser and ExplicitPassword fields. This option is only supported for Microsoft CAs. The</td></tr>
</table>

Name	Description
	default is *false*. **Tip:** This option is set to *true* primarily for Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest.
SubscriberTerms	A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (true) or not (false). The default is *false*. **Tip:** Configure a link to the custom terms using the *URL to Subscriber Terms* application setting. See Application Settings: Enrollment Tab on page 609 for more information.
ExplicitUser	A string indicating the username, in the format DOMAIN\username, for a service account user in the forest in which the Microsoft CA resides or, for non-domain-joined machines, local machine account credentials on the machine on which the CA is installed when *ExplicitCredentials* is set to *true*. **Tip:** This service account user needs appropriate permissions in the Microsoft CA security settings to accomplish the tasks you plan to carry out for this CA through Keyfactor Command. For example: • Certificate enrollment • Certificate revocation • Certificate key recovery • Certificate request approval and denial These tasks will be carried out on the CA in the context of the credentials you provide here. Access control for these tasks on CAs is controlled with Keyfactor Command security (see Security Roles and Claims on page 622) and the *AllowedRequesters* option. **Note:** When the *ExplicitCredentials* option is configured, enrollment and other tasks (e.g. revocation) is done in the context of the user configured here, not the user making the request in Keyfactor Command. This overrides the existing AD security policy used by Keyfactor Command.
ExplicitPassword	An object indicating the password information to use for authentication

Name	Description
	along with the *ExplicitUser* if *ExplicitCredentials* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
UseAllowedRequesters	A Boolean that sets whether the allowed requesters option is enabled (*true*) or not (*false*). See also *AllowedRequesters*. The default is *false*. **Tip:** This option is supported for all CAs, but it must be used for Microsoft CAs where integrated authentication is not supported and EJBCA CAs. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA. **Tip:** For CAs in a two-way trust you don't usually need to enable *UseAllowedRequesters* on the CA, though this may be required in some circumstances depending on the security configuration in the environment. However, templates for a two-way trust environment always require configuration of this option at a template level to support enrollment (see Certificate Template Operations on page 381 and see PUT Templates on page 2435).

AllowedRequesters	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example:
	`"AllowedRequesters": [`
	`  "Power Users",`

Name	Description
	<pre> "Read Only"]</pre> The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA. This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting *request certificates* for the selected security roles at the CA level on a Microsoft CA. In addition to granting permissions at the CA level, you need to enable the *UseAllowedRequesters* option to grant permissions on a template-by-template basis (see PUT Templates on page 2435). The values set here are only considered if *UseAllowedRequesters* is set to *true*.
FullScan	An object indicating the schedule for the full synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported: **Name** / **Description** Off — Turn off a previously configured schedule.

Name	Description

	Name / **Description**
	Interval — A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
	Name / **Description**
	Minutes — An integer indicating the number of minutes between each interval.
	For example, every hour:
	`"Interval": { "Minutes": 60 }`
	Daily — A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
	Name / **Description**
	Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
	For example, daily at 11:30 pm:
	`"Daily": { "Time": "2023-11-25T23:30:00Z" }`
	Weekly — A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

Name	Description
	Name / **Description**

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Days	An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).

For example, every Monday, Wednesday and Friday at 5:30 pm:

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

Tip: There are two types of synchronization schedules available for CAs—Full and Incremental. You do not necessarily need to configure both types. A full scan reads all the certificates and certificate requests in the CA database and synchronizes them to Keyfactor Command regardless of their current state in Keyfactor Command. An incremental scan reads the certificates and certificate requests in the CA database that have been generated since the last full or incremental scan and synchronizes them to Keyfactor Command. A

Name	Description
	common configuration would be a full scan once or twice a week to provide a clean image of the CA database with a frequent incremental scan to provide timely updates to Keyfactor Command. For a large CA database, a full scan can take a long time to complete. Since an incremental scan only synchronizes updates that have occurred to the CA database since the last synchronization was run, this process is generally quick (other than for the initial synchronization when Keyfactor Command is first installed). The frequency of the incremental scans would depend on the volume of certificate requests coming into the CA.
IncrementalScan	An object indicating the schedule for the incremental synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
   "Minutes": 60
}
```

</td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ThresholdCheck</td><td>An object indicating the schedule for threshold monitoring checks on this certificate authority (see MonitorThresholds). The following schedule types are supported:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>CAType</td><td>An integer indicating the type of CA:
0—DCOM
1—HTTPS</td></tr>
<tr><td>AuthCertificatePassword</td><td>An object indicating the password for the certificate to use to authenticate to the EJBCA CA.
Due to its sensitive nature, this value is not returned in responses.</td></tr>
<tr><td>AuthCertificate</td><td>An object containing information about the client certificate used to provide authentication to the EJBCA CA. This certificate is used to authenticate to the EJBCA database for synchronization, enrollment and management of certificates.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Authentication certificate values include:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>IssuedDN</td><td>A string indicating the distinguished name of the client certificate used to authenticate to the EJBCA CA in X.500 format. For example:
<pre>"IssuedDN":
"CN=SuperAdmin,OU=IT,O="\Key
Example,
Inc.\",L=Independence,ST=OH,C=US"</pre></td></tr><tr><td>IssuerDN</td><td>A string indicating the distinguished name of the EJBCA CA in X.500 format.</td></tr><tr><td>Thumbprint</td><td>A string indicating the thumbprint of the client certificate used to authenticate to the EJBCA CA.</td></tr><tr><td>ExpirationDate</td><td>A string indicating the expiration date of the client certificate used to authenticate to the EJBCA CA.</td></tr></table></td></tr>
<tr><td>EnforceUniqueDN</td><td>A Boolean that sets whether the unique DN requirement is enforced on the CA (true) or not (false).
Checking this will cause Keyfactor Command, upon enrollment, to search the EJBCA CA for end entities with DNs that match the DN in the certificate request. If a matching DN is found, the process will update the existing end entity in EJBCA with the new certificate request information rather than creating a new end entity. If you enable this option in Keyfactor Command, it must also be enabled on the matching EJBCA CA. A mismatch in these settings can result in enrollment failures.
This setting applies to HTTPS CAs only.</td></tr>
<tr><td>AllowOneClickRenewals</td><td>A Boolean that sets whether the CA will allow (true) One-Click Renewal on certificates in this CA or not (false). If you wish to use One-Click Renewal for certificates, the Allow One-Click Renewals option must be enabled in both the templates and CAs to which you want One-Click Renewal to apply (see <a>Certificate Template Operations on page 381 and <a>Adding or Modifying a CA Record on page 354).</td></tr>
<tr><td>NewEndEntityOnRe-newAndReissue</td><td>A Boolean that sets whether a renewal and reissue requests against an EJBCA CA will create a new end entity for the new certificate (true) or attempt to associate the new certificate with the existing end entity (false).</td></tr>
</table>

Name	Description
	The default is *false*. This value must be set to *true* for CA records created for the Keyfactor AnyCA Gateway REST to allow renewal and reissue enrollments to succeed.
LastScan	A string indicating the date, in UTC, on which a synchronization was last performed for the CA.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.3 GET Certificate Authority

The GET /CertificateAuthority method is used to retrieve a list of certificate authorities defined in Keyfactor Command. This method returns HTTP 200 OK on a success with details for all the defined certificate authorities.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 264: GET Certificate Authority Input Parameters

Name	In	Description
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 265: GET Certificate Authority Response Data

Name	Description
Id	An integer indicating the Keyfactor Command identifier for the certificate authority. The ID is automatically assigned by Keyfactor Command.
LogicalName	A string indicating the logical name of the certificate authority.
HostName	A string indicating the DNS hostname (for DCOM configurations) or URL (for HTTPS configurations) of the certificate authority (e.g. myca.keyexample.com or https://myca.keyexample.com).
Delegate	A Boolean that sets whether management interactions with the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, these interactions are done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
DelegateEnrollment	A Boolean that sets whether enrollment to the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, enrollment is done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
ForestRoot	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). **Note:** This field is retained for legacy purposes and will auto-populate with the value provided in the *ConfigurationTenant* field.
ConfigurationTenant	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). For EJBCA CAs, this is a reference ID and does not need to be the DNS domain name. The short hostname of the EJBCA CA server makes a good reference ID.

Name	Description
	Important: EJBCA and Microsoft CAs cannot be configured with the same *Configuration Tenant*, so do not set this to the DNS domain name for an EJBCA CA if you will also be configuring Microsoft CAs in the same DNS domain.
Remote	A Boolean that sets whether communications with the certificate authority are done via a Keyfactor Universal Orchestrator configured to manage remote CAs. If set to *true*, a value must be provided for the *Agent*. The default is *false*.
Agent	A string indicating the GUID of the Keyfactor Universal Orchestrator configured to manage the certificate authority (see *Remote*).
Standalone	A Boolean that sets whether the certificate authority is a standalone CA (*true*) or not (*false*). If both *Standalone* is set to *true* and *AllowedEnrollmentTypes* is set to 1 or 3, *KeyRetention* may be set. The default is *false*.
MonitorThresholds	A Boolean that sets whether threshold monitoring is enabled. If set to *true*, email alerts will be sent when certificate issuance or failures (including denials) since the last threshold alert was sent falls outside the configured limits. If this option is set to *true*, the following additional fields should also be set: • IssuanceMax • IssuanceMin • FailureMax The DenialMax field has been deprecated and should always be zero. Monitoring is not supported for CAs accessed with the Keyfactor Universal Orchestrator. The default is *false*. See also *ThresholdCheck* to configure the monitoring frequency. **Note:** For full functionality of threshold monitoring, you must also configure email recipients for threshold alerts. These are configured globally rather than on a CA-by-CA basis. See Certificate Authority Monitoring on page 378 for more information.
IssuanceMax	An integer that sets the maximum number of certificates that can be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.

Name	Description
IssuanceMin	An integer that sets the minimum number of certificates that should be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If fewer certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
DenialMax	This is considered deprecated and may be removed in a future release.
FailureMax	An integer that sets the maximum number of certificate requests that can fail or be denied in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificate requests than this fail in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
RFCEnforcement	A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (*true*) or not (*false*). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. The default is *false*. **Tip:** The *RFCEnforcement* option at the CA level is used only for standalone CAs. RFC enforcement for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435).
Properties	A string indicating additional properties about the certificate authority. This field is used to store the configuration for the *Sync External Certificates* option. This option allows foreign certificates that have been imported into a Microsoft CA to be synchronized to Keyfactor Command along with the certificates issued by the Microsoft CA. The setting is referenced using the following format: `{\"syncExternal\":true} OR {\"syncExternal\":false}`
AllowedEnrollmentTypes	An integer that sets the type(s) of enrollment that are allowed through Keyfactor Command for the certificate authority. Possible values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>PFX and CSR Enrollment</td></tr>
</table>
This value is unset by default.
</td></tr>
<tr><td>KeyRetention</td><td>
An integer that sets the type of key retention to enable for the certificate authority, if any. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Key Retention Disabled</td></tr>
<tr><td>1</td><td>Indefinite</td></tr>
<tr><td>2</td><td>After Expiration</td></tr>
<tr><td>3</td><td>From Issuance</td></tr>
</table>
Values of 2 and 3 require setting KeyRetentionDays.
This value is unset by default.
Tip: The KeyRetention option at the CA level is used only for standalone CAs. Key retention for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435). KeyRetention on a CA may only be set to a value other than zero if both Standalone is set to true and AllowedEnrollmentTypes is set to 1 or 3. Some level of private key retention must be configured when using PFX enrollment with a standalone CA. See Adding or Modifying a CA Record on page 354 for more information.
</td></tr>
<tr><td>KeyRetentionDays</td><td>An integer indicating the number of days for which to retain the private keys for certificates issued by this certificate authority before scheduling them for deletion. This value is unset by default.</td></tr>
<tr><td>ExplicitCredentials</td><td>A Boolean that sets whether explicit credentials are enabled for this certificate authority (true) or not (false). Set this to true for CAs that do not support integrated authentication or are not configured for integrated authentication and enter credentials in the ExplicitUser and ExplicitPassword fields. This option is only supported for Microsoft CAs. The</td></tr>
</table>

Name	Description
	default is *false*. **Tip:** This option is set to *true* primarily for Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest.
SubscriberTerms	A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (true) or not (false). The default is *false*. **Tip:** Configure a link to the custom terms using the *URL to Subscriber Terms* application setting. See Application Settings: Enrollment Tab on page 609 for more information.
ExplicitUser	A string indicating the username, in the format DOMAIN\username, for a service account user in the forest in which the Microsoft CA resides or, for non-domain-joined machines, local machine account credentials on the machine on which the CA is installed when *ExplicitCredentials* is set to *true*. **Tip:** This service account user needs appropriate permissions in the Microsoft CA security settings to accomplish the tasks you plan to carry out for this CA through Keyfactor Command. For example: • Certificate enrollment • Certificate revocation • Certificate key recovery • Certificate request approval and denial These tasks will be carried out on the CA in the context of the credentials you provide here. Access control for these tasks on CAs is controlled with Keyfactor Command security (see Security Roles and Claims on page 622) and the *AllowedRequesters* option. **Note:** When the *ExplicitCredentials* option is configured, enrollment and other tasks (e.g. revocation) is done in the context of the user configured here, not the user making the request in Keyfactor Command. This overrides the existing AD security policy used by Keyfactor Command.
ExplicitPassword	An object indicating the password information to use for authentication

Name	Description
	along with the *ExplicitUser* if *ExplicitCredentials* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
UseAllowedRequesters	A Boolean that sets whether the allowed requesters option is enabled (*true*) or not (*false*). See also *AllowedRequesters*. The default is *false*. **Tip:** This option is supported for all CAs, but it must be used for Microsoft CAs where integrated authentication is not supported and EJBCA CAs. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA. **Tip:** For CAs in a two-way trust you don't usually need to enable *UseAllowedRequesters* on the CA, though this may be required in some circumstances depending on the security configuration in the environment. However, templates for a two-way trust environment always require configuration of this option at a template level to support enrollment (see Certificate Template Operations on page 381 and see PUT Templates on page 2435).

AllowedRequesters	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example:
	`"AllowedRequesters": [`
	`"Power Users",`

Name	Description

	<pre>"Read Only"
	]</pre>
	The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA.
	This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting *request certificates* for the selected security roles at the CA level on a Microsoft CA.
	In addition to granting permissions at the CA level, you need to enable the *UseAllowedRequesters* option to grant permissions on a template-by-template basis (see PUT Templates on page 2435).
	The values set here are only considered if *UseAllowedRequesters* is set to *true*.

FullScan	An object indicating the schedule for the full synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported: **Name** / **Description** Off — Turn off a previously configured schedule.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Days	An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).

For example, every Monday, Wednesday and Friday at 5:30 pm:

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

Tip: There are two types of synchronization schedules available for CAs—Full and Incremental. You do not necessarily need to configure both types. A full scan reads all the certificates and certificate requests in the CA database and synchronizes them to Keyfactor Command regardless of their current state in Keyfactor Command. An incremental scan reads the certificates and certificate requests in the CA database that have been generated since the last full or incremental scan and synchronizes them to Keyfactor Command. A

Name	Description
	common configuration would be a full scan once or twice a week to provide a clean image of the CA database with a frequent incremental scan to provide timely updates to Keyfactor Command. For a large CA database, a full scan can take a long time to complete. Since an incremental scan only synchronizes updates that have occurred to the CA database since the last synchronization was run, this process is generally quick (other than for the initial synchronization when Keyfactor Command is first installed). The frequency of the incremental scans would depend on the volume of certificate requests coming into the CA.
IncrementalScan	An object indicating the schedule for the incremental synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>
   "Minutes": 60
}
</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ThresholdCheck</td><td>An object indicating the schedule for threshold monitoring checks on this certificate authority (see MonitorThresholds). The following schedule types are supported:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>CAType</td><td>An integer indicating the type of CA:
0—DCOM
1—HTTPS</td></tr>
<tr><td>AuthCertificatePassword</td><td>An object indicating the password for the certificate to use to authenticate to the EJBCA CA.
Due to its sensitive nature, this value is not returned in responses.</td></tr>
<tr><td>AuthCertificate</td><td>An object containing information about the client certificate used to provide authentication to the EJBCA CA. This certificate is used to authenticate to the EJBCA database for synchronization, enrollment and management of certificates.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Authentication certificate values include:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>IssuedDN</td><td>A string indicating the distinguished name of the client certificate used to authenticate to the EJBCA CA in X.500 format. For example:
<pre>"IssuedDN":
"CN=SuperAdmin,OU=IT,O="\Key
Example,
Inc.\",L=Independence,ST=OH,C=US"</pre></td></tr><tr><td>IssuerDN</td><td>A string indicating the distinguished name of the EJBCA CA in X.500 format.</td></tr><tr><td>Thumbprint</td><td>A string indicating the thumbprint of the client certificate used to authenticate to the EJBCA CA.</td></tr><tr><td>ExpirationDate</td><td>A string indicating the expiration date of the client certificate used to authenticate to the EJBCA CA.</td></tr></table></td></tr>
<tr><td>EnforceUniqueDN</td><td>A Boolean that sets whether the unique DN requirement is enforced on the CA (true) or not (false).
Checking this will cause Keyfactor Command, upon enrollment, to search the EJBCA CA for end entities with DNs that match the DN in the certificate request. If a matching DN is found, the process will update the existing end entity in EJBCA with the new certificate request information rather than creating a new end entity. If you enable this option in Keyfactor Command, it must also be enabled on the matching EJBCA CA. A mismatch in these settings can result in enrollment failures.
This setting applies to HTTPS CAs only.</td></tr>
<tr><td>AllowOneClickRenewals</td><td>A Boolean that sets whether the CA will allow (true) One-Click Renewal on certificates in this CA or not (false). If you wish to use One-Click Renewal for certificates, the Allow One-Click Renewals option must be enabled in both the templates and CAs to which you want One-Click Renewal to apply (see <a>Certificate Template Operations on page 381 and <a>Adding or Modifying a CA Record on page 354).</td></tr>
<tr><td>NewEndEntityOnRe-newAndReissue</td><td>A Boolean that sets whether a renewal and reissue requests against an EJBCA CA will create a new end entity for the new certificate (true) or attempt to associate the new certificate with the existing end entity (false).</td></tr>
</table>

Name	Description
	The default is *false*. This value must be set to *true* for CA records created for the Keyfactor AnyCA Gateway REST to allow renewal and reissue enrollments to succeed.
LastScan	A string indicating the date, in UTC, on which a synchronization was last performed for the CA.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.4 POST Certificate Authority

The POST /CertificateAuthority method is used to create a new certificate authority record in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the CA configuration.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Table 266: POST Certificate Authority Input Parameters

Name	In	Description
LogicalName	Body	**Required**. A string indicating the logical name of the certificate authority.
HostName	Body	**Required**. A string indicating the DNS hostname (for DCOM configurations) or URL (for HTTPS configurations) of the certificate authority (e.g. myca.keyexample.com or https://-myca.keyexample.com).
Delegate	Body	A Boolean that sets whether management interactions with the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, these interactions are done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
DelegateEnrollment	Body	A Boolean that sets whether enrollment to the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, enrollment is done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
ForestRoot	Body	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). **Note:** This field is retained for legacy purposes and will auto-populate with the value provided in the *ConfigurationTenant* field.
ConfigurationTenant	Body	**Required**. A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). For EJBCA CAs, this is a reference ID and does not need to be the DNS domain name. The short hostname of the EJBCA CA server

Name	In	Description
		makes a good reference ID. **Important:** EJBCA and Microsoft CAs cannot be configured with the same *Configuration Tenant*, so do not set this to the DNS domain name for an EJBCA CA if you will also be configuring Microsoft CAs in the same DNS domain.
Remote	Body	A Boolean that sets whether communications with the certificate authority are done via a Keyfactor Universal Orchestrator configured to manage remote CAs. If set to *true*, a value must be provided for the *Agent*. The default is *false*.
Agent	Body	A string indicating the GUID of the Keyfactor Universal Orchestrator configured to manage the certificate authority (see *Remote*).
Standalone	Body	A Boolean that sets whether the certificate authority is a standalone CA (*true*) or not (*false*). If both *Standalone* is set to *true* and *AllowedEnrollmentTypes* is set to 1 or 3, *KeyRetention* may be set. The default is *false*.
MonitorThresholds	Body	A Boolean that sets whether threshold monitoring is enabled. If set to *true*, email alerts will be sent when certificate issuance or failures (including denials) since the last threshold alert was sent falls outside the configured limits. If this option is set to *true*, the following additional fields should also be set: • IssuanceMax • IssuanceMin • FailureMax The DenialMax field has been deprecated and should always be zero. Monitoring is not supported for CAs accessed with the Keyfactor Universal Orchestrator. The default is *false*. See also *ThresholdCheck* to configure the monitoring frequency. **Note:** For full functionality of threshold monitoring, you must also configure email recipients for threshold alerts. These are configured globally rather than on a CA-by-CA basis. See Certificate Authority Monitoring on page 378 for more information.
IssuanceMax	Body	An integer that sets the maximum number of certificates that can be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificates than this are

Name	In	Description
		issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
IssuanceMin	Body	An integer that sets the minimum number of certificates that should be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If fewer certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
DenialMax	Body	This is considered deprecated and may be removed in a future release.
FailureMax	Body	An integer that sets the maximum number of certificate requests that can fail or be denied in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificate requests than this fail in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
RFCEnforcement	Body	A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (*true*) or not (*false*). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. The default is *false*. **Tip:** The *RFCEnforcement* option at the CA level is used only for standalone CAs. RFC enforcement for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435).
Properties	Body	**Required**. A string indicating additional properties about the certificate authority. This field is used to store the configuration for the *Sync External Certificates* option. This option allows foreign certificates that have been imported into a Microsoft CA to be synchronized to Keyfactor Command along with the certificates issued by the Microsoft CA. The setting is referenced using the following format: `{\"syncExternal\":true} OR {\"syncExternal\":false}`
AllowedEnrollmentTypes	Body	An integer that sets the type(s) of enrollment that are allowed through Keyfactor Command for the certificate authority. Possible values are:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>PFX and CSR Enrollment</td></tr>
</table>
This value is unset by default.
</td></tr>
<tr><td>KeyRetention</td><td>Body</td><td>An integer that sets the type of key retention to enable for the certificate authority, if any. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Key Retention Disabled</td></tr>
<tr><td>1</td><td>Indefinite</td></tr>
<tr><td>2</td><td>After Expiration</td></tr>
<tr><td>3</td><td>From Issuance</td></tr>
</table>
Values of 2 and 3 require setting KeyRetentionDays.
This value is unset by default.

Tip: The KeyRetention option at the CA level is used only for standalone CAs. Key retention for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435).
KeyRetention on a CA may only be set to a value other than zero if both Standalone is set to true and AllowedEnrollmentTypes is set to 1 or 3. Some level of private key retention must be configured when using PFX enrollment with a standalone CA. See Adding or Modifying a CA Record on page 354 for more information.</td></tr>
<tr><td>KeyRetentionDays</td><td>Body</td><td>An integer indicating the number of days for which to retain the private keys for certificates issued by this certificate authority before scheduling them for deletion. This value is unset by default.</td></tr>
<tr><td>ExplicitCredentials</td><td>Body</td><td>A Boolean that sets whether explicit credentials are enabled for this certificate authority (true) or not (false). Set this to true for CAs that</td></tr>
</table>

Name	In	Description
		do not support integrated authentication or are not configured for integrated authentication and enter credentials in the *ExplicitUser* and *ExplicitPassword* fields. This option is only supported for Microsoft CAs. The default is *false*. **Tip:** This option is set to *true* primarily for Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest.
SubscriberTerms	Body	A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (true) or not (false). The default is *false*. **Tip:** Configure a link to the custom terms using the *URL to Subscriber Terms* application setting. See Application Settings: Enrollment Tab on page 609 for more information.
ExplicitUser	Body	A string indicating the username, in the format DOMAIN\username, for a service account user in the forest in which the Microsoft CA resides or, for non-domain-joined machines, local machine account credentials on the machine on which the CA is installed when *ExplicitCredentials* is set to *true*. **Tip:** This service account user needs appropriate permissions in the Microsoft CA security settings to accomplish the tasks you plan to carry out for this CA through Keyfactor Command. For example: • Certificate enrollment • Certificate revocation • Certificate key recovery • Certificate request approval and denial These tasks will be carried out on the CA in the context of the credentials you provide here. Access control for these tasks on CAs is controlled with Keyfactor Command security (see Security Roles and Claims on page 622) and the *AllowedRequesters* option. **Note:** When the *ExplicitCredentials* option is configured,

Name	In	Description
		enrollment and other tasks (e.g. revocation) is done in the context of the user configured here, not the user making the request in Keyfactor Command. This overrides the existing AD security policy used by Keyfactor Command.
ExplicitPassword	Body	An object indicating the password information to use for authentication along with the *ExplicitUser* if *ExplicitCredentials* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
UseAllowedRequesters	Body	A Boolean that sets whether the allowed requesters option is enabled (*true*) or not (*false*). See also *AllowedRequesters*. The default is *false*. **Tip:** This option is supported for all CAs, but it must be used for Microsoft CAs where integrated authentication is not supported and EJBCA CAs. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA. **Tip:** For CAs in a two-way trust you don't usually need to enable *UseAllowedRequesters* on the CA, though this may be required in some circumstances depending on the security configuration in the environment. However, templates for a two-way trust environment always require configuration of this option at a template level to support enrollment (see Certi-

Name	In	Description
		ficate Template Operations on page 381 and see PUT Templates on page 2435).
AllowedRequesters	Body	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example: `"AllowedRequesters": [ "Power Users", "Read Only" ]` The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA. This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting *request certificates* for the selected security roles at the CA level on a Microsoft CA. In addition to granting permissions at the CA level, you need to enable the *UseAllowedRequesters* option to grant permissions on a template-by-template basis (see PUT Templates on page 2435). The values set here are only considered if *UseAllowedRequesters* is set to *true*.
FullScan	Body	An object indicating the schedule for the full synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>were created via the API.
The following schedule types are supported:
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interv-al</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:m-m:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"</pre></td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td></td><td><pre>}</pre></td></tr>
<tr><td>Week-ly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:m-m:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for
</td></tr>
</table>

Name	In	Description

		this endpoint.
		
		For example:
		
		`"FullScan": { "Daily": { "Time": "2022-05-27T17:30:00Z" } }`
		
		Or:
		
		`"FullScan": { "Weekly": { "Days": [ "Monday", "Wednesday", "Friday" ], "Time": "2022-05-27T17:30:00Z" } }`
		
		Tip: There are two types of synchronization schedules available for CAs—Full and Incremental. You do not necessarily need to configure both types. A full scan reads all the certificates and certificate requests in the CA database and synchronizes them to Keyfactor Command regardless of their current state in Keyfactor Command. An incremental scan reads the certificates and certificate requests in the CA database that have been generated since the last full or incremental scan and synchronizes them to Keyfactor Command. A common configuration would be a full scan once or twice a week to provide a clean image of the CA database with a frequent incremental scan to provide timely updates to Keyfactor Command. For a large CA database, a full scan can take a long time to complete. Since an incremental scan only synchronizes updates that have occurred to the CA database since the last synchronization was run, this process is generally quick (other than for the initial synchronization when Keyfactor Command is first installed). The frequency of the incremental scans would depend on the volume of certificate requests coming into the CA.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>IncrementalScan</td><td>Body</td><td>An object indicating the schedule for the incremental synchronization of this certificate authority.
Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don’t meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API.
The following schedule types are supported:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr><tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr></table>
For example, every hour:
<pre>"Interval": {
  "Minutes": 60
}</pre></td></tr><tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ThresholdCheck</td><td>Body</td><td>
An object indicating the schedule for threshold monitoring checks on this certificate authority (see MonitorThresholds). The following schedule types are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>
A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>CAType</td><td>Body</td><td>An integer indicating the type of CA:
0—DCOM
1—HTTPS</td></tr>
<tr><td>AuthCertificatePassword</td><td>Body</td><td>An object indicating the password for the certificate to use to authenticate to the EJBCA CA.
Supported methods to store certificate and associated password information are:
Store the credential information in the Keyfactor secrets table.</td></tr>
</table>

Name	In	Description

		A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.
		• Load the credential information from a PAM provider.
		See Privileged Access Management (PAM) on page 742 and PAM Providers on page 1871 for more information.
		
		Value / **Description**
		SecretValue — A string containing the password used to security the EJBCA CA client authentication certificate.
		Parameters — An object indicating the parameters to supply for PAM authentication. These will vary depending on the PAM provider.
		Provider — A string indicating the ID of the PAM provider. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the ID.
		
		For example, the password stored as a Keyfactor secret will look like:
		`{ "SecretValue": "MySuperSecretPassword" }`
		
		The password stored as a CyberArk PAM secret will look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898 and the Folder and Object reference the folder name and object name in the CyberArk safe):
		`{ "Provider": "1", "Parameters":{ "Folder":"MyFolderName", "Object":"MyEJBCAClientAuthPassword" } }`

Name	In	Description
		The password stored as a Delinea PAM secret will look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898 and the SecretId is the ID if the secret created in the Delinea secret server for this purpose): `{ "Provider": "1", "Parameters":{ "SecretId":"MyEJBCAPasswordId" } }` Due to its sensitive nature, this value is not returned in responses.
AuthCertificate	Body	An object containing information about the client certificate used to provide authentication to the EJBCA CA. This certificate is used to authenticate to the EJBCA database for synchronization, enrollment and management of certificates. The syntax is the same as for *AuthCertificatePassword* with the *SecretValue*. The *SecretValue* is set to a string containing the base-64-encoded representation of the certificate with private key (in PKCS#12 format) that will be used to authenticate to the EJBCA CA. For example: `{ "SecretValue": "MIACAQMwgAYJKoZIhvcNAQcBoIAkgASCA+[truncated for display]gQUwRndGMbmlkmwIOuC0MbOY1EyDpACAwGQAAAA" }` Due to its sensitive nature, this value is not returned in this format in responses.
EnforceUniqueDN	Body	A Boolean that sets whether the unique DN requirement is enforced on the CA (*true*) or not (*false*). Checking this will cause Keyfactor Command, upon enrollment, to search the EJBCA CA for end entities with DNs that match the DN in the certificate request. If a matching DN is found, the process will update the existing end entity in EJBCA with the new certificate request information rather than creating a new end entity. If you enable this option in Keyfactor Command, it must also be enabled on the matching EJBCA CA. A mismatch in these settings can result in enrollment failures. This setting applies to HTTPS CAs only.

Name	In	Description
AllowOneClickRenewals	Body	A Boolean that sets whether the CA will allow (true) *One-Click Renewal* on certificates in this CA or not (false). If you wish to use *One-Click Renewal* for certificates, the **Allow One-Click Renewals** option must be enabled in both the templates and CAs to which you want *One-Click Renewal* to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354).
NewEndEntityOnRenewAndReissue	Body	A Boolean that sets whether a renewal and reissue requests against an EJBCA CA will create a new end entity for the new certificate (true) or attempt to associate the new certificate with the existing end entity (false). The default is *false*. This value must be set to *true* for CA records created for the Keyfactor AnyCA Gateway REST to allow renewal and reissue enrollments to succeed.
LastScan	Body	A string indicating the date, in UTC, on which a synchronization was last performed for the CA.

Table 267: POST Certificate Authority Response Data

Name	Description
Id	An integer indicating the Keyfactor Command identifier for the certificate authority. The ID is automatically assigned by Keyfactor Command.
LogicalName	A string indicating the logical name of the certificate authority.
HostName	A string indicating the DNS hostname (for DCOM configurations) or URL (for HTTPS configurations) of the certificate authority (e.g. myca.keyexample.com or https://myca.keyexample.com).
Delegate	A Boolean that sets whether management interactions with the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, these interactions are done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
DelegateEnrollment	A Boolean that sets whether enrollment to the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, enrollment is done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
ForestRoot	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). **Note:** This field is retained for legacy purposes and will auto-populate with the value provided in the *ConfigurationTenant* field.
ConfigurationTenant	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). For EJBCA CAs, this is a reference ID and does not need to be the DNS domain name. The short hostname of the EJBCA CA server makes a good reference ID.

Name	Description
	Important: EJBCA and Microsoft CAs cannot be configured with the same *Configuration Tenant*, so do not set this to the DNS domain name for an EJBCA CA if you will also be configuring Microsoft CAs in the same DNS domain.
Remote	A Boolean that sets whether communications with the certificate authority are done via a Keyfactor Universal Orchestrator configured to manage remote CAs. If set to *true*, a value must be provided for the *Agent*. The default is *false*.
Agent	A string indicating the GUID of the Keyfactor Universal Orchestrator configured to manage the certificate authority (see *Remote*).
Standalone	A Boolean that sets whether the certificate authority is a standalone CA (*true*) or not (*false*). If both *Standalone* is set to *true* and *AllowedEnrollmentTypes* is set to 1 or 3, *KeyRetention* may be set. The default is *false*.
MonitorThresholds	A Boolean that sets whether threshold monitoring is enabled. If set to *true*, email alerts will be sent when certificate issuance or failures (including denials) since the last threshold alert was sent falls outside the configured limits. If this option is set to *true*, the following additional fields should also be set: • IssuanceMax • IssuanceMin • FailureMax The DenialMax field has been deprecated and should always be zero. Monitoring is not supported for CAs accessed with the Keyfactor Universal Orchestrator. The default is *false*. See also *ThresholdCheck* to configure the monitoring frequency. **Note:** For full functionality of threshold monitoring, you must also configure email recipients for threshold alerts. These are configured globally rather than on a CA-by-CA basis. See Certificate Authority Monitoring on page 378 for more information.
IssuanceMax	An integer that sets the maximum number of certificates that can be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.

Name	Description
IssuanceMin	An integer that sets the minimum number of certificates that should be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If fewer certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
DenialMax	This is considered deprecated and may be removed in a future release.
FailureMax	An integer that sets the maximum number of certificate requests that can fail or be denied in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificate requests than this fail in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
RFCEnforcement	A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (*true*) or not (*false*). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. The default is *false*. **Tip:** The *RFCEnforcement* option at the CA level is used only for standalone CAs. RFC enforcement for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435).
Properties	A string indicating additional properties about the certificate authority. This field is used to store the configuration for the *Sync External Certificates* option. This option allows foreign certificates that have been imported into a Microsoft CA to be synchronized to Keyfactor Command along with the certificates issued by the Microsoft CA. The setting is referenced using the following format: `{\"syncExternal\":true} OR {\"syncExternal\":false}`
AllowedEnrollmentTypes	An integer that sets the type(s) of enrollment that are allowed through Keyfactor Command for the certificate authority. Possible values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>PFX and CSR Enrollment</td></tr>
</table>
This value is unset by default.
</td></tr>
<tr><td>KeyRetention</td><td>An integer that sets the type of key retention to enable for the certificate authority, if any. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Key Retention Disabled</td></tr>
<tr><td>1</td><td>Indefinite</td></tr>
<tr><td>2</td><td>After Expiration</td></tr>
<tr><td>3</td><td>From Issuance</td></tr>
</table>
Values of 2 and 3 require setting KeyRetentionDays.
This value is unset by default.

Tip: The KeyRetention option at the CA level is used only for standalone CAs. Key retention for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435). KeyRetention on a CA may only be set to a value other than zero if both Standalone is set to true and AllowedEnrollmentTypes is set to 1 or 3. Some level of private key retention must be configured when using PFX enrollment with a standalone CA. See Adding or Modifying a CA Record on page 354 for more information.</td></tr>
<tr><td>KeyRetentionDays</td><td>An integer indicating the number of days for which to retain the private keys for certificates issued by this certificate authority before scheduling them for deletion. This value is unset by default.</td></tr>
<tr><td>ExplicitCredentials</td><td>A Boolean that sets whether explicit credentials are enabled for this certificate authority (true) or not (false). Set this to true for CAs that do not support integrated authentication or are not configured for integrated authentication and enter credentials in the ExplicitUser and ExplicitPassword fields. This option is only supported for Microsoft CAs. The</td></tr>
</table>

Name	Description
	default is *false*. **Tip:** This option is set to *true* primarily for Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest.
SubscriberTerms	A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (true) or not (false). The default is *false*. **Tip:** Configure a link to the custom terms using the *URL to Subscriber Terms* application setting. See Application Settings: Enrollment Tab on page 609 for more information.
ExplicitUser	A string indicating the username, in the format DOMAIN\username, for a service account user in the forest in which the Microsoft CA resides or, for non-domain-joined machines, local machine account credentials on the machine on which the CA is installed when *ExplicitCredentials* is set to *true*. **Tip:** This service account user needs appropriate permissions in the Microsoft CA security settings to accomplish the tasks you plan to carry out for this CA through Keyfactor Command. For example: • Certificate enrollment • Certificate revocation • Certificate key recovery • Certificate request approval and denial These tasks will be carried out on the CA in the context of the credentials you provide here. Access control for these tasks on CAs is controlled with Keyfactor Command security (see Security Roles and Claims on page 622) and the *AllowedRequesters* option. **Note:** When the *ExplicitCredentials* option is configured, enrollment and other tasks (e.g. revocation) is done in the context of the user configured here, not the user making the request in Keyfactor Command. This overrides the existing AD security policy used by Keyfactor Command.
ExplicitPassword	An object indicating the password information to use for authentication

Name	Description
	along with the *ExplicitUser* if *ExplicitCredentials* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
UseAllowedRequesters	A Boolean that sets whether the allowed requesters option is enabled (*true*) or not (*false*). See also *AllowedRequesters*. The default is *false*. **Tip:** This option is supported for all CAs, but it must be used for Microsoft CAs where integrated authentication is not supported and EJBCA CAs. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA. **Tip:** For CAs in a two-way trust you don't usually need to enable *UseAllowedRequesters* on the CA, though this may be required in some circumstances depending on the security configuration in the environment. However, templates for a two-way trust environment always require configuration of this option at a template level to support enrollment (see Certificate Template Operations on page 381 and see PUT Templates on page 2435).

AllowedRequesters	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example:
	`"AllowedRequesters": [`
	`  "Power Users",`

Name	Description

	<pre>"Read Only"
	]</pre>
	The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA.
	This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting *request certificates* for the selected security roles at the CA level on a Microsoft CA.
	In addition to granting permissions at the CA level, you need to enable the *UseAllowedRequesters* option to grant permissions on a template-by-template basis (see PUT Templates on page 2435).
	The values set here are only considered if *UseAllowedRequesters* is set to *true*.

FullScan	An object indicating the schedule for the full synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported: **Name** / **Description** Off — Turn off a previously configured schedule.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:</td></tr>
</table>
</td></tr>
</table>

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr><tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr></table> For example, every Monday, Wednesday and Friday at 5:30 pm: <pre>"Weekly": {
	  "Days": [
	    "Monday",
	    "Wednesday",
	    "Friday"
	  ],
	  "Time": "2023-11-27T17:30:00Z"
	}</pre></td></tr></table>
	  **Note:** Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
	  **Tip:** There are two types of synchronization schedules available for CAs—Full and Incremental. You do not necessarily need to configure both types. A full scan reads all the certificates and certificate requests in the CA database and synchronizes them to Keyfactor Command regardless of their current state in Keyfactor Command. An incremental scan reads the certificates and certificate requests in the CA database that have been generated since the last full or incremental scan and synchronizes them to Keyfactor Command. A

Name	Description
	common configuration would be a full scan once or twice a week to provide a clean image of the CA database with a frequent incremental scan to provide timely updates to Keyfactor Command. For a large CA database, a full scan can take a long time to complete. Since an incremental scan only synchronizes updates that have occurred to the CA database since the last synchronization was run, this process is generally quick (other than for the initial synchronization when Keyfactor Command is first installed). The frequency of the incremental scans would depend on the volume of certificate requests coming into the CA.
IncrementalScan	An object indicating the schedule for the incremental synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ThresholdCheck</td><td>An object indicating the schedule for threshold monitoring checks on this certificate authority (see MonitorThresholds). The following schedule types are supported:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>CAType</td><td>An integer indicating the type of CA:
0—DCOM
1—HTTPS</td></tr>
<tr><td>AuthCertificatePassword</td><td>An object indicating the password for the certificate to use to authenticate to the EJBCA CA.
Due to its sensitive nature, this value is not returned in responses.</td></tr>
<tr><td>AuthCertificate</td><td>An object containing information about the client certificate used to provide authentication to the EJBCA CA. This certificate is used to authenticate to the EJBCA database for synchronization, enrollment and management of certificates.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Authentication certificate values include:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>IssuedDN</td><td>A string indicating the distinguished name of the client certificate used to authenticate to the EJBCA CA in X.500 format. For example:
<pre>"IssuedDN":
"CN=SuperAdmin,OU=IT,O="\Key
Example,
Inc.\",L=Independence,ST=OH,C=US"</pre></td></tr><tr><td>IssuerDN</td><td>A string indicating the distinguished name of the EJBCA CA in X.500 format.</td></tr><tr><td>Thumbprint</td><td>A string indicating the thumbprint of the client certificate used to authenticate to the EJBCA CA.</td></tr><tr><td>ExpirationDate</td><td>A string indicating the expiration date of the client certificate used to authenticate to the EJBCA CA.</td></tr></table></td></tr>
<tr><td>EnforceUniqueDN</td><td>A Boolean that sets whether the unique DN requirement is enforced on the CA (true) or not (false).
Checking this will cause Keyfactor Command, upon enrollment, to search the EJBCA CA for end entities with DNs that match the DN in the certificate request. If a matching DN is found, the process will update the existing end entity in EJBCA with the new certificate request information rather than creating a new end entity. If you enable this option in Keyfactor Command, it must also be enabled on the matching EJBCA CA. A mismatch in these settings can result in enrollment failures.
This setting applies to HTTPS CAs only.</td></tr>
<tr><td>AllowOneClickRenewals</td><td>A Boolean that sets whether the CA will allow (true) One-Click Renewal on certificates in this CA or not (false). If you wish to use One-Click Renewal for certificates, the Allow One-Click Renewals option must be enabled in both the templates and CAs to which you want One-Click Renewal to apply (see <a>Certificate Template Operations on page 381 and <a>Adding or Modifying a CA Record on page 354).</td></tr>
<tr><td>NewEndEntityOnRe-newAndReissue</td><td>A Boolean that sets whether a renewal and reissue requests against an EJBCA CA will create a new end entity for the new certificate (true) or attempt to associate the new certificate with the existing end entity (false).</td></tr>
</table>

Name	Description
	The default is *false*. This value must be set to *true* for CA records created for the Keyfactor AnyCA Gateway REST to allow renewal and reissue enrollments to succeed.
LastScan	A string indicating the date, in UTC, on which a synchronization was last performed for the CA.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.5 PUT Certificate Authority

The PUT /CertificateAuthority method is used to update a certificate authority record in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the CA configuration.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 268: PUT Certificate Authority Input Parameters

Name	In	Description
Id	Body	**Required**. An integer indicating the Keyfactor Command identifier for the certificate authority. The ID is automatically assigned by Keyfactor Command.
LogicalName	Body	**Required**. A string indicating the logical name of the certificate authority.
HostName	Body	**Required**. A string indicating the DNS hostname (for DCOM configurations) or URL (for HTTPS configurations) of the certificate authority (e.g. myca.keyexample.com or https://-myca.keyexample.com).
Delegate	Body	A Boolean that sets whether management interactions with the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, these interactions are done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
DelegateEnrollment	Body	A Boolean that sets whether enrollment to the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, enrollment is done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
ForestRoot	Body	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). **Note:** This field is retained for legacy purposes and will auto-populate with the value provided in the *ConfigurationTenant* field.

Name	In	Description
ConfigurationTenant	Body	**Required**. A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). For EJBCA CAs, this is a reference ID and does not need to be the DNS domain name. The short hostname of the EJBCA CA server makes a good reference ID. **Important:** EJBCA and Microsoft CAs cannot be configured with the same *Configuration Tenant*, so do not set this to the DNS domain name for an EJBCA CA if you will also be configuring Microsoft CAs in the same DNS domain.
Remote	Body	A Boolean that sets whether communications with the certificate authority are done via a Keyfactor Universal Orchestrator configured to manage remote CAs. If set to *true*, a value must be provided for the *Agent*. The default is *false*.
Agent	Body	A string indicating the GUID of the Keyfactor Universal Orchestrator configured to manage the certificate authority (see *Remote*).
Standalone	Body	A Boolean that sets whether the certificate authority is a standalone CA (*true*) or not (*false*). If both *Standalone* is set to *true* and *AllowedEnrollmentTypes* is set to 1 or 3, *KeyRetention* may be set. The default is *false*.
MonitorThresholds	Body	A Boolean that sets whether threshold monitoring is enabled. If set to *true*, email alerts will be sent when certificate issuance or failures (including denials) since the last threshold alert was sent falls outside the configured limits. If this option is set to *true*, the following additional fields should also be set: • IssuanceMax • IssuanceMin • FailureMax The DenialMax field has been deprecated and should always be zero. Monitoring is not supported for CAs accessed with the Keyfactor Universal Orchestrator. The default is *false*. See also *ThresholdCheck* to configure the monitoring frequency. **Note:** For full functionality of threshold monitoring, you must also configure email recipients for threshold alerts. These are configured globally rather than on a CA-by-CA basis. See Certificate Authority Monitoring on page 378 for more inform-

Name	In	Description
		ation.
IssuanceMax	Body	An integer that sets the maximum number of certificates that can be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
IssuanceMin	Body	An integer that sets the minimum number of certificates that should be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If fewer certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
DenialMax	Body	This is considered deprecated and may be removed in a future release.
FailureMax	Body	An integer that sets the maximum number of certificate requests that can fail or be denied in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificate requests than this fail in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
RFCEnforcement	Body	A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (*true*) or not (*false*). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. The default is *false*. **Tip:** The *RFCEnforcement* option at the CA level is used only for standalone CAs. RFC enforcement for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435).
Properties	Body	**Required**. A string indicating additional properties about the certificate authority. This field is used to store the configuration for the *Sync External Certificates* option. This option allows foreign certificates that have been imported into a Microsoft CA to be synchronized to Keyfactor Command along with the certificates issued by the Microsoft CA. The setting is referenced using the following format:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td><code>{\"syncExternal\":true} OR {\"syncExternal\":false}</code></td></tr>
<tr><td>AllowedEnrollmentTypes</td><td>Body</td><td>An integer that sets the type(s) of enrollment that are allowed through Keyfactor Command for the certificate authority. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>PFX Enrollment</td></tr><tr><td>2</td><td>CSR Enrollment</td></tr><tr><td>3</td><td>PFX and CSR Enrollment</td></tr></table>This value is unset by default.</td></tr>
<tr><td>KeyRetention</td><td>Body</td><td>An integer that sets the type of key retention to enable for the certificate authority, if any. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Key Retention Disabled</td></tr><tr><td>1</td><td>Indefinite</td></tr><tr><td>2</td><td>After Expiration</td></tr><tr><td>3</td><td>From Issuance</td></tr></table>Values of 2 and 3 require setting KeyRetentionDays.
This value is unset by default.
Tip: The KeyRetention option at the CA level is used only for standalone CAs. Key retention for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435).
KeyRetention on a CA may only be set to a value other than zero if both Standalone is set to true and AllowedEnrollmentTypes is set to 1 or 3. Some level of private key retention must be configured when using PFX enrollment with a standalone CA. See Adding or Modifying a CA Record on page 354 for more information.</td></tr>
<tr><td>KeyRetentionDays</td><td>Body</td><td>An integer indicating the number of days for which to retain the</td></tr>
</table>

Name	In	Description
	y	private keys for certificates issued by this certificate authority before scheduling them for deletion. This value is unset by default.
ExplicitCredentials	Body	A Boolean that sets whether explicit credentials are enabled for this certificate authority (*true*) or not (*false*). Set this to *true* for CAs that do not support integrated authentication or are not configured for integrated authentication and enter credentials in the *ExplicitUser* and *ExplicitPassword* fields. This option is only supported for Microsoft CAs. The default is *false*. **Tip:** This option is set to *true* primarily for Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest.
SubscriberTerms	Body	A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (true) or not (false). The default is *false*. **Tip:** Configure a link to the custom terms using the *URL to Subscriber Terms* application setting. See Application Settings: Enrollment Tab on page 609 for more information.
ExplicitUser	Body	A string indicating the username, in the format DOMAIN\username, for a service account user in the forest in which the Microsoft CA resides or, for non-domain-joined machines, local machine account credentials on the machine on which the CA is installed when *ExplicitCredentials* is set to *true*. **Tip:** This service account user needs appropriate permissions in the Microsoft CA security settings to accomplish the tasks you plan to carry out for this CA through Keyfactor Command. For example: • Certificate enrollment • Certificate revocation • Certificate key recovery • Certificate request approval and denial These tasks will be carried out on the CA in the context of the credentials you provide here. Access control for these tasks on CAs is controlled with Keyfactor Command security (see

Name	In	Description
		Security Roles and Claims on page 622) and the *AllowedRequesters* option. **Note:** When the *ExplicitCredentials* option is configured, enrollment and other tasks (e.g. revocation) is done in the context of the user configured here, not the user making the request in Keyfactor Command. This overrides the existing AD security policy used by Keyfactor Command.
ExplicitPassword	Body	An object indicating the password information to use for authentication along with the *ExplicitUser* if *ExplicitCredentials* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
UseAllowedRequesters	Body	A Boolean that sets whether the allowed requesters option is enabled (*true*) or not (*false*). See also *AllowedRequesters*. The default is *false*. **Tip:** This option is supported for all CAs, but it must be used for Microsoft CAs where integrated authentication is not supported and EJBCA CAs. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA. **Tip:** For CAs in a two-way trust you don't usually need to

Name	In	Description
		enable *UseAllowedRequesters* on the CA, though this may be required in some circumstances depending on the security configuration in the environment. However, templates for a two-way trust environment always require configuration of this option at a template level to support enrollment (see Certificate Template Operations on page 381 and see PUT Templates on page 2435).
AllowedRequesters	Body	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example: `"AllowedRequesters": [ "Power Users", "Read Only" ]` The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA. This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting *request certificates* for the selected security roles at the CA level on a Microsoft CA. In addition to granting permissions at the CA level, you need to enable the *UseAllowedRequesters* option to grant permissions on a template-by-template basis (see PUT Templates on page 2435). The values set here are only considered if *UseAllowedRequesters* is set to *true*.
FullScan	Body	An object indicating the schedule for the full synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don’t meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API.
The following schedule types are supported:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr><tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr></table>
For example, every hour:
<pre>"Interval": {
  "Minutes": 60
}</pre></td></tr><tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:m-</td></tr></table></td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>m:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
<tr><td>Week-ly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:m-m:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",</pre>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description

Nam-e	Description

```
        "Wednesday",
        "Friday"
    ],
    "Time": "2023-11-27T17:30:00Z"
}
```

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

For example:

```
"FullScan": {
   "Daily": {
      "Time": "2022-05-27T17:30:00Z"
   }
}
```

Or:

```
"FullScan": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"
      ],
      "Time": "2022-05-27T17:30:00Z"
   }
}
```

Tip: There are two types of synchronization schedules available for CAs—Full and Incremental. You do not necessarily need to configure both types. A full scan reads all the certificates and certificate requests in the CA database and synchronizes them to Keyfactor Command regardless of their

Name	In	Description
		current state in Keyfactor Command. An incremental scan reads the certificates and certificate requests in the CA database that have been generated since the last full or incremental scan and synchronizes them to Keyfactor Command. A common configuration would be a full scan once or twice a week to provide a clean image of the CA database with a frequent incremental scan to provide timely updates to Keyfactor Command. For a large CA database, a full scan can take a long time to complete. Since an incremental scan only synchronizes updates that have occurred to the CA database since the last synchronization was run, this process is generally quick (other than for the initial synchronization when Keyfactor Command is first installed). The frequency of the incremental scans would depend on the volume of certificate requests coming into the CA.
IncrementalScan	Body	An object indicating the schedule for the incremental synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported: **Name** / **Description** Off — Turn off a previously configured schedule. Interval — A dictionary that indicates a job scheduled to run every **Name** / **Description**

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr></table>For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Week-ly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr></table></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ThresholdCheck</td><td>Bod-y</td><td>An object indicating the schedule for threshold monitoring checks on this certificate authority (see MonitorThresholds). The following schedule types are supported:
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available
</td></tr>
</table>

Name	In	Description
		in the Management Portal for this functionality—are valid for this endpoint.
CAType	Body	An integer indicating the type of CA: • 0—DCOM • 1—HTTPS
AuthCertificatePassword	Body	An object indicating the password for the certificate to use to authenticate to the EJBCA CA. Supported methods to store certificate and associated password information are: • Store the credential information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the credential information from a PAM provider. See Privileged Access Management (PAM) on page 742 and PAM Providers on page 1871 for more information.

Value	Description
SecretValue	A string containing the password used to security the EJBCA CA client authentication certificate.
Parameters	An object indicating the parameters to supply for PAM authentication. These will vary depending on the PAM provider.
Provider	A string indicating the ID of the PAM provider. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the ID.

For example, the password stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "MySuperSecretPassword"
}
```

Name	In	Description

		The password stored as a CyberArk PAM secret will look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898 and the Folder and Object reference the folder name and object name in the CyberArk safe):
		
		`{ "Provider": "1", "Parameters":{ "Folder":"MyFolderName", "Object":"MyEJBCAClientAuthPassword" } }`
		
		The password stored as a Delinea PAM secret will look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898 and the SecretId is the ID if the secret created in the Delinea secret server for this purpose):
		
		`{ "Provider": "1", "Parameters":{ "SecretId":"MyEJBCAPasswordId" } }`
		
		Due to its sensitive nature, this value is not returned in responses.

AuthCertificate	Body	An object containing information about the client certificate used to provide authentication to the EJBCA CA. This certificate is used to authenticate to the EJBCA database for synchronization, enrollment and management of certificates. The syntax is the same as for *AuthCertificatePassword* with the *SecretValue*. The *SecretValue* is set to a string containing the base-64-encoded representation of the certificate with private key (in PKCS#12 format) that will be used to authenticate to the EJBCA CA. For example: `{ "SecretValue": "MIACAQMwgAYJKoZIhvcNAQcBoIAkgASCA+[truncated for display]gQUwRndGMbmlkmwIOuC0MbOY1EyDpACAwGQAAAA" }` Due to its sensitive nature, this value is not returned in this format in responses.

Name	In	Description
EnforceUniqueDN	Body	A Boolean that sets whether the unique DN requirement is enforced on the CA (*true*) or not (*false*). Checking this will cause Keyfactor Command, upon enrollment, to search the EJBCA CA for end entities with DNs that match the DN in the certificate request. If a matching DN is found, the process will update the existing end entity in EJBCA with the new certificate request information rather than creating a new end entity. If you enable this option in Keyfactor Command, it must also be enabled on the matching EJBCA CA. A mismatch in these settings can result in enrollment failures. This setting applies to HTTPS CAs only.
AllowOneClickRenewals	Body	A Boolean that sets whether the CA will allow (true) *One-Click Renewal* on certificates in this CA or not (false). If you wish to use *One-Click Renewal* for certificates, the **Allow One-Click Renewals** option must be enabled in both the templates and CAs to which you want *One-Click Renewal* to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354).
NewEndEntityOnRenewAndReissue	Body	A Boolean that sets whether a renewal and reissue requests against an EJBCA CA will create a new end entity for the new certificate (true) or attempt to associate the new certificate with the existing end entity (false). The default is *false*. This value must be set to *true* for CA records created for the Keyfactor AnyCA Gateway REST to allow renewal and reissue enrollments to succeed.
LastScan	Body	A string indicating the date, in UTC, on which a synchronization was last performed for the CA.

Table 269: PUT Certificate Authority Response Data

Name	Description
Id	An integer indicating the Keyfactor Command identifier for the certificate authority. The ID is automatically assigned by Keyfactor Command.
LogicalName	A string indicating the logical name of the certificate authority.
HostName	A string indicating the DNS hostname (for DCOM configurations) or URL (for HTTPS configurations) of the certificate authority (e.g. myca.keyexample.com or https://myca.keyexample.com).
Delegate	A Boolean that sets whether management interactions with the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, these interactions are done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
DelegateEnrollment	A Boolean that sets whether enrollment to the certificate authority via Keyfactor Command should be done in the context of the user making the request (*true*). If set to *false*, enrollment is done in the context of the service account under which the Keyfactor Command application pool is running unless *ExplicitCredentials* is true. **Important:** Delegation is only supported with Microsoft CAs and has limitations. Be sure to read more about delegation in Adding or Modifying a CA Record on page 354 before setting this option to true.
ForestRoot	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). **Note:** This field is retained for legacy purposes and will auto-populate with the value provided in the *ConfigurationTenant* field.
ConfigurationTenant	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). For EJBCA CAs, this is a reference ID and does not need to be the DNS domain name. The short hostname of the EJBCA CA server makes a good reference ID.

Name	Description
	Important: EJBCA and Microsoft CAs cannot be configured with the same *Configuration Tenant*, so do not set this to the DNS domain name for an EJBCA CA if you will also be configuring Microsoft CAs in the same DNS domain.
Remote	A Boolean that sets whether communications with the certificate authority are done via a Keyfactor Universal Orchestrator configured to manage remote CAs. If set to *true*, a value must be provided for the *Agent*. The default is *false*.
Agent	A string indicating the GUID of the Keyfactor Universal Orchestrator configured to manage the certificate authority (see *Remote*).
Standalone	A Boolean that sets whether the certificate authority is a standalone CA (*true*) or not (*false*). If both *Standalone* is set to *true* and *AllowedEnrollmentTypes* is set to 1 or 3, *KeyRetention* may be set. The default is *false*.
MonitorThresholds	A Boolean that sets whether threshold monitoring is enabled. If set to *true*, email alerts will be sent when certificate issuance or failures (including denials) since the last threshold alert was sent falls outside the configured limits. If this option is set to *true*, the following additional fields should also be set: • IssuanceMax • IssuanceMin • FailureMax The DenialMax field has been deprecated and should always be zero. Monitoring is not supported for CAs accessed with the Keyfactor Universal Orchestrator. The default is *false*. See also *ThresholdCheck* to configure the monitoring frequency. **Note:** For full functionality of threshold monitoring, you must also configure email recipients for threshold alerts. These are configured globally rather than on a CA-by-CA basis. See Certificate Authority Monitoring on page 378 for more information.
IssuanceMax	An integer that sets the maximum number of certificates that can be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.

Name	Description
IssuanceMin	An integer that sets the minimum number of certificates that should be issued in the period between scheduled threshold monitoring alert emails before an alert is triggered. If fewer certificates than this are issued in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
DenialMax	This is considered deprecated and may be removed in a future release.
FailureMax	An integer that sets the maximum number of certificate requests that can fail or be denied in the period between scheduled threshold monitoring alert emails before an alert is triggered. If more certificate requests than this fail in the period, a notification will be included in the threshold monitoring email. This value is unset by default.
RFCEnforcement	A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (*true*) or not (*false*). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. The default is *false*. **Tip:** The *RFCEnforcement* option at the CA level is used only for standalone CAs. RFC enforcement for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435).
Properties	A string indicating additional properties about the certificate authority. This field is used to store the configuration for the *Sync External Certificates* option. This option allows foreign certificates that have been imported into a Microsoft CA to be synchronized to Keyfactor Command along with the certificates issued by the Microsoft CA. The setting is referenced using the following format: `{\"syncExternal\":true} OR {\"syncExternal\":false}`
AllowedEnrollmentTypes	An integer that sets the type(s) of enrollment that are allowed through Keyfactor Command for the certificate authority. Possible values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>PFX and CSR Enrollment</td></tr>
</table>
This value is unset by default.
</td></tr>
<tr><td>KeyRetention</td><td>
An integer that sets the type of key retention to enable for the certificate authority, if any. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Key Retention Disabled</td></tr>
<tr><td>1</td><td>Indefinite</td></tr>
<tr><td>2</td><td>After Expiration</td></tr>
<tr><td>3</td><td>From Issuance</td></tr>
</table>
Values of 2 and 3 require setting KeyRetentionDays.
This value is unset by default.
Tip: The KeyRetention option at the CA level is used only for standalone CAs. Key retention for enterprise CAs is configured on a template-by-template basis (see PUT Templates on page 2435). KeyRetention on a CA may only be set to a value other than zero if both Standalone is set to true and AllowedEnrollmentTypes is set to 1 or 3. Some level of private key retention must be configured when using PFX enrollment with a standalone CA. See Adding or Modifying a CA Record on page 354 for more information.
</td></tr>
<tr><td>KeyRetentionDays</td><td>An integer indicating the number of days for which to retain the private keys for certificates issued by this certificate authority before scheduling them for deletion. This value is unset by default.</td></tr>
<tr><td>ExplicitCredentials</td><td>A Boolean that sets whether explicit credentials are enabled for this certificate authority (true) or not (false). Set this to true for CAs that do not support integrated authentication or are not configured for integrated authentication and enter credentials in the ExplicitUser and ExplicitPassword fields. This option is only supported for Microsoft CAs. The</td></tr>
</table>

Name	Description
	default is *false*. **Tip:** This option is set to *true* primarily for Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest.
SubscriberTerms	A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (true) or not (false). The default is *false*. **Tip:** Configure a link to the custom terms using the *URL to Subscriber Terms* application setting. See Application Settings: Enrollment Tab on page 609 for more information.
ExplicitUser	A string indicating the username, in the format DOMAIN\username, for a service account user in the forest in which the Microsoft CA resides or, for non-domain-joined machines, local machine account credentials on the machine on which the CA is installed when *ExplicitCredentials* is set to *true*. **Tip:** This service account user needs appropriate permissions in the Microsoft CA security settings to accomplish the tasks you plan to carry out for this CA through Keyfactor Command. For example: • Certificate enrollment • Certificate revocation • Certificate key recovery • Certificate request approval and denial These tasks will be carried out on the CA in the context of the credentials you provide here. Access control for these tasks on CAs is controlled with Keyfactor Command security (see Security Roles and Claims on page 622) and the *AllowedRequesters* option. **Note:** When the *ExplicitCredentials* option is configured, enrollment and other tasks (e.g. revocation) is done in the context of the user configured here, not the user making the request in Keyfactor Command. This overrides the existing AD security policy used by Keyfactor Command.
ExplicitPassword	An object indicating the password information to use for authentication

Name	Description
	along with the *ExplicitUser* if *ExplicitCredentials* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
UseAllowedRequesters	A Boolean that sets whether the allowed requesters option is enabled (*true*) or not (*false*). See also *AllowedRequesters*. The default is *false*. **Tip:** This option is supported for all CAs, but it must be used for Microsoft CAs where integrated authentication is not supported and EJBCA CAs. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA. **Tip:** For CAs in a two-way trust you don't usually need to enable *UseAllowedRequesters* on the CA, though this may be required in some circumstances depending on the security configuration in the environment. However, templates for a two-way trust environment always require configuration of this option at a template level to support enrollment (see Certificate Template Operations on page 381 and see PUT Templates on page 2435).

AllowedRequesters	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example:
	`"AllowedRequesters": [`
	`"Power Users",`

Name	Description

	<pre>"Read Only"
	]</pre>
	The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA.
	This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting *request certificates* for the selected security roles at the CA level on a Microsoft CA.
	In addition to granting permissions at the CA level, you need to enable the *UseAllowedRequesters* option to grant permissions on a template-by-template basis (see PUT Templates on page 2435).
	The values set here are only considered if *UseAllowedRequesters* is set to *true*.

FullScan	An object indicating the schedule for the full synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported: **Name** / **Description** Off — Turn off a previously configured schedule.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Days	An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).

For example, every Monday, Wednesday and Friday at 5:30 pm:

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

Tip: There are two types of synchronization schedules available for CAs—Full and Incremental. You do not necessarily need to configure both types. A full scan reads all the certificates and certificate requests in the CA database and synchronizes them to Keyfactor Command regardless of their current state in Keyfactor Command. An incremental scan reads the certificates and certificate requests in the CA database that have been generated since the last full or incremental scan and synchronizes them to Keyfactor Command. A

Name	Description
	common configuration would be a full scan once or twice a week to provide a clean image of the CA database with a frequent incremental scan to provide timely updates to Keyfactor Command. For a large CA database, a full scan can take a long time to complete. Since an incremental scan only synchronizes updates that have occurred to the CA database since the last synchronization was run, this process is generally quick (other than for the initial synchronization when Keyfactor Command is first installed). The frequency of the incremental scans would depend on the volume of certificate requests coming into the CA.
IncrementalScan	An object indicating the schedule for the incremental synchronization of this certificate authority. Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API. The following schedule types are supported:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>
   "Minutes": 60
}
</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ThresholdCheck</td><td>An object indicating the schedule for threshold monitoring checks on this certificate authority (see MonitorThresholds). The following schedule types are supported:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>CAType</td><td>An integer indicating the type of CA:
0—DCOM
1—HTTPS</td></tr>
<tr><td>AuthCertificatePassword</td><td>An object indicating the password for the certificate to use to authenticate to the EJBCA CA.
Due to its sensitive nature, this value is not returned in responses.</td></tr>
<tr><td>AuthCertificate</td><td>An object containing information about the client certificate used to provide authentication to the EJBCA CA. This certificate is used to authenticate to the EJBCA database for synchronization, enrollment and management of certificates.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Authentication certificate values include:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>IssuedDN</td><td>A string indicating the distinguished name of the client certificate used to authenticate to the EJBCA CA in X.500 format. For example:
<pre>"IssuedDN":
"CN=SuperAdmin,OU=IT,O="\Key
Example,
Inc.\",L=Independence,ST=OH,C=US"</pre></td></tr><tr><td>IssuerDN</td><td>A string indicating the distinguished name of the EJBCA CA in X.500 format.</td></tr><tr><td>Thumbprint</td><td>A string indicating the thumbprint of the client certificate used to authenticate to the EJBCA CA.</td></tr><tr><td>ExpirationDate</td><td>A string indicating the expiration date of the client certificate used to authenticate to the EJBCA CA.</td></tr></table></td></tr>
<tr><td>EnforceUniqueDN</td><td>A Boolean that sets whether the unique DN requirement is enforced on the CA (true) or not (false).
Checking this will cause Keyfactor Command, upon enrollment, to search the EJBCA CA for end entities with DNs that match the DN in the certificate request. If a matching DN is found, the process will update the existing end entity in EJBCA with the new certificate request information rather than creating a new end entity. If you enable this option in Keyfactor Command, it must also be enabled on the matching EJBCA CA. A mismatch in these settings can result in enrollment failures.
This setting applies to HTTPS CAs only.</td></tr>
<tr><td>AllowOneClickRenewals</td><td>A Boolean that sets whether the CA will allow (true) One-Click Renewal on certificates in this CA or not (false). If you wish to use One-Click Renewal for certificates, the Allow One-Click Renewals option must be enabled in both the templates and CAs to which you want One-Click Renewal to apply (see <a>Certificate Template Operations on page 381 and <a>Adding or Modifying a CA Record on page 354).</td></tr>
<tr><td>NewEndEntityOnRe-newAndReissue</td><td>A Boolean that sets whether a renewal and reissue requests against an EJBCA CA will create a new end entity for the new certificate (true) or attempt to associate the new certificate with the existing end entity (false).</td></tr>
</table>

Name	Description
	The default is *false*. This value must be set to *true* for CA records created for the Keyfactor AnyCA Gateway REST to allow renewal and reissue enrollments to succeed.
LastScan	A string indicating the date, in UTC, on which a synchronization was last performed for the CA.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.6 POST Certificate Authority Test

The POST /CertificateAuthority/Test method is used to validate that a connection can be made to the certificate authority with the provided information. This method returns HTTP 200 OK on a success with details for the success or failure of the CA validation.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 270: POST Certificate Authority Test Input Parameters

Name	In	Description
id	Body	**Required**. An integer indicating the CA id in the Keyfactor database.
CAType	Body	An integer indicating the type of CA: • 0—DCOM Use this option for Microsoft CAs and CA gateways. • 1—HTTPS Use this option for EJBCA CAs. The default is 0.
ExplicitCredentials	Body	A Boolean that sets whether explicit credentials are enabled for this certificate authority (*true*) or not (*false*). Set this to *true* for CAs that do not support integrated authentication or are not configured for integrated authentication and enter credentials in the *ExplicitUser* and *ExplicitPassword* fields. This option is only supported for Microsoft CAs. The default is *false*. **Tip:** This option is set to *true* primarily for Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest.

ExplicitPassword	Body	A string containing either *null*, or the password for the *ExplicitUser*, including: SecretValue, parameters, and provider if applicable.
		`"ExplicitePassword": {`
		`"secretValue": "string",`
		`"parameters": {`
		`"additionalProp1": "string",`
		`"additionalProp2": "string",`
		`"additionalProp3": "string"`
		`},`
		`"provider": 0`

ExplicitUser	Body	A string indicating the username, in the format DOMAIN\username, for a service account user in the forest in which the

Name	In	Description
		Microsoft CA resides or, for non-domain-joined machines, local machine account credentials on the machine on which the CA is installed when *ExplicitCredentials* is set to *true*. **Tip:** This service account user needs appropriate permissions in the Microsoft CA security settings to accomplish the tasks you plan to carry out for this CA through Keyfactor Command. For example: • Certificate enrollment • Certificate revocation • Certificate key recovery • Certificate request approval and denial These tasks will be carried out on the CA in the context of the credentials you provide here. Access control for these tasks on CAs is controlled with Keyfactor Command security (see Security Roles and Claims on page 622) and the *AllowedRequesters* option. **Note:** When the *ExplicitCredentials* option is configured, enrollment and other tasks (e.g. revocation) is done in the context of the user configured here, not the user making the request in Keyfactor Command. This overrides the existing AD security policy used by Keyfactor Command.

AuthCertificate	Body	**Required***. An object indicating the PKCS#12 client certificate to use to authenticate to the CA using the following format. This certificate is used to authenticate to the CA database for synchronization, enrollment and management of certificates. The certificate is provided in the following format:
		
		`"AuthCertificate": {`
		`"secretValue": "string",`
		`"parameters": {`
		`"IssuedDN": "CN=superadmincert",`
		`"IssuerDN": "CN=CorpIssuingCA1, DC=keyexample, DC=com",`
		`"Thumbprint": "913D80B33517DD6F42428664883DA43BB64D0EEE",`
		`"ExpirationDate": "2025-07-17T18:24:23Z"`
		`},`
		`"provider": 0`

Name	In	Description
		This parameter is required for EJBCA CAs.
AuthCertificatePassword	Body	**Required***. An object containing password for the client certificate used to provide authentication to the CA. `"authCertificatePassword": { "secretValue": "string", "parameters": { "additionalProp1": "string", "additionalProp2": "string", "additionalProp3": "string" }, "provider": 0` This parameter is required for EJBCA CAs.
LogicalName	Body	**Required**. A string indicating the logical name of the certificate authority.
HostName	Body	**Required**. A string indicating the DNS hostname (for DCOM configurations) or URL (for HTTPS configurations) of the certificate authority (e.g. myca.keyexample.com or https://-myca.keyexample.com).
ConfigurationTenant	Body	**Required***. A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). This parameter is required for Microsoft CAs.
ForestRoot	Body	A string indicating the forest root name or DNS domain name for the certificate authority (e.g. keyexample.com). **Note:** This field is retained for legacy purposes and will auto-populate with the value provided in the *ConfigurationTenant* field.

Table 271: POST Certificate Authority Test Response Data

Name	Description
Success	A Boolean that indicates whether the CA could successfully be reached (True) or not (False).
Message	A string indicating a message about the validation test of the certificate authority.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.7 POST Certificate Authority PublishCRL

The POST /CertificateAuthority/PublishCRL method is used to publish a Certificate Revocation List from a specified Certificate Authority to its defined publication points. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/revoke/
OR
/certificates/collections/revoke/#/ (where # is a reference to a specific certificate collectionPAM provider ID)

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 272: POST Certificate Authority PublishCRL Input Parameters

Name	In	Description
CertificateAuthorityHostName	Body	The host name of the machine hosting the CA. This field is optional, but is recommended.
CertificateAuthorityLogicalName	Body	**Required**. The logical name of the CA.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.8 GET Certificate Authority Source Count

The GET /CertificateAuthority/SourceCount method is used to retrieve the count of certificate authorities with full or incremental synchronization scans configured. This method returns HTTP 200 OK on a success with a count. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 273: GET Certificate Authority Source Count Response Body

Name	Description
n/a	An integer indicating the number of CAs with with full or incremental synchronization scans configured in the Keyfactor Command database.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.9 GET Certificate Authority Available Forests

The GET /CertificateAuthority/AvailableForests method is used to retrieve the list of Active Directory forests that are available to Keyfactor Command (the current forest and any forests in a two-way trust with this forest). This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 274: GET Certificate Authority Available Forests Response Body

Name	Description
n/a	An array of strings containing the list of the available Active Directory forests.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.10 GET Certificate Authority Health Monitoring Schedule

The GET /CertificateAuthority/HealthMonitoring/Schedule method is used to retrieve the current schedule for the CA health monitoring job This method returns HTTP 200 OK on a success with the list of schedule settings. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 275: GET Certificate Authority Health Monitoring Schedule Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the schedule.
Schedule	A string indicating the schedule for the health monitoring job. Schedules are shown in cron syntax. For an interval schedule, this will look like I_mm where mm is the number of minutes (e.g. I_30 for every 30 minutes). For daily schedules, this will look like D_hh:mm where hh:mm is the time to run the job (e.g. D_14:30 for daily at 2:30 pm).
ScheduleType	An integer indicating the type of schedule. Health monitoring schedules have type 10.
Enabled	A Boolean that indicates whether health monitoring is enabled (true) or not (false).
Name	A string indicating the Keyfactor Command reference name of the health monitoring job. This is the name that appears in log output for the job.
EntityId	An internally used Keyfactor Command field.
LastRun	A string indicating the last run time of the job in ISO 8601 UTC time format (e.g. 2023-05-19T16:23:01Z).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.11 GET Certificate Authority Alert Recipients CA Health Recipients

The GET /CertificateAuthority/AlertRecipients/CAHealthRecipients method is used to retrieve the list of recipients configured in Keyfactor Command for CA health monitoring alerts. This method

returns HTTP 200 OK on a success with the list of CA health recipients. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 276: GET Certificate Authority Alert Recipients CA Health Recipients Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the recipient.
Email	A string indicating the email address of the recipient.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.12 POST Certificate Authority Alert Recipients CA Health Recipients

The POST /CertificateAuthority/AlertRecipients/CAHealthRecipients method is used to create new recipients to receive CA health monitoring alerts in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the health monitoring recipients.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Table 277: POST Certificate Authority Alert Recipients CA Health Recipients Input Body

Name	In	Description
Emails	Body	**Required**. An object containing a set of strings with the email address of each recipient. For example: `{ "Emails": [ "Recipient1@keyexample.com","Recipient3@keyexample.com" ] }`

Table 278: POST Certificate Authority Alert Recipients CA Health Recipients Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the recipient.
Email	A string indicating the email address of the recipient. **Note:** Only recipients created with the POST request are returned in the response. Any pre-existing recipients are not included in the response.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.13 GET Certificate Authority Alert Recipients CA Health Recipients ID

The GET /CertificateAuthority/AlertRecipients/CAHealthRecipients/{id} method is used to retrieve the CA health monitoring recipient configured in Keyfactor Command with the specified ID. This method returns HTTP 200 OK on a success with the details for the recipient.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 279: GET Certificate Authority Alert Recipients CA Health Recipients {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer that specifies the Keyfactor Command reference ID of the certificate authority health monitoring recipient record to retrieve. Use the *GET /CertificateAuthority/AlertRecipients/CAHealthRecipients* method (see GET Certificate Authority Alert Recipients CA Health Recipients on page 1286) to retrieve the ID.

Table 280: GET Certificate Authority Alert Recipients CA Health Recipients {id} Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the recipient.
Email	A string indicating the email address of the recipient.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.14 DELETE Certificate Authority Alert Recipients CA Health Recipients ID

The DELETE /CertificateAuthority/AlertRecipients/CAHealthRecipients/{id} endpoint is used to delete the CA health monitoring recipient with the specified Keyfactor Command reference ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Table 281: DELETE Certificate Authority Alert Recipients CA Health Recipients {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer that specifies the Keyfactor Command reference ID of the certificate authority health monitoring recipient record to delete. Use the *GET /CertificateAuthority/AlertRecipients/CAHealthRecipients* method (see GET Certificate Authority Alert Recipients CA Health Recipients on page 1286) to retrieve the ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.15 PUT Certificate Authority Alert Recipients CA Health Recipients ID

The PUT /CertificateAuthority/AlertRecipients/CAHealthRecipients/{id} method is used to update the CA health monitoring alert recipient with the specified ID. This method returns HTTP 200 OK on a success with the details submitted.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Table 282: PUT Certificate Authority Alert Recipients CA Health Recipients {id} Input Body

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the CA health monitoring recipient. Use the *GET /CertificateAuthority/AlertRecipients/CAHealthRecipients* method (see GET Certificate Authority Alert Recipients CA Health Recipients on page 1286) to retrieve the ID.
Email	Body	**Required**. A string indicating the updated email address of the recipient.

Table 283: PUT Certificate Authority Alert Recipients CA Health Recipients {id} Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the recipient.
Email	A string indicating the new email address of the recipient.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.16 DELETE Certificate Authority Alert Recipients CA Threshold Recipients ID

The DELETE /CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id} endpoint is used to delete the CA threshold recipient with the specified Keyfactor Command reference ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/certificate_authorities/modify/

Table 284: DELETE Certificate Authority Alert Recipients CA Threshold Recipient {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer that specifies the Keyfactor Command reference ID of the certificate authority threshold recipient record to delete. Use the *GET /CertificateAuthority/AlertRecipients/CAThesholdRecipients* method (see GET Certificate Authority Alert Recipients CA Threshold Recipients below) to retrieve the ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.17 GET Certificate Authority Alert Recipients CA Threshold Recipients

The GET /CertificateAuthority/AlertRecipients/CAThresholdRecipients method is used to retrieve the list of recipients configured in Keyfactor Command for CA threshold alerts. This method returns HTTP 200 OK on a success with the list of threshold alert recipients. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 285: GET Certificate Authority Alert Recipients CA Threshold Recipients Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the recipient.
Email	A string indicating the email address of the recipient.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.18 GET Certificate Authority Alert Recipients CA Threshold Recipients ID

The GET /CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id} method is used to retrieve the CA threshold recipient configured in Keyfactor Command with the specified ID. This method returns HTTP 200 OK on a success with the details of the CA threshold recipient.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/read/

Table 286: GET Certificate Authority Alert Recipients CA Threshold Recipients {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer that specifies the reference ID of the certificate authority threshold recipient record to retrieve. Use the *GET /CertificateAuthority/AlertRecipients/CAThesholdRecipients* method (see GET Certificate Authority Alert Recipients CA Threshold Recipients on the previous page) to retrieve the ID.

Table 287: GET Certificate Authority Alert Recipients CA Threshold Recipients {id} Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the CA threshold recipient.
Email	A string indicating the email address of the CA threshold recipient.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.19 POST Certificate Authority Alert Recipients CA Threshold Recipients

The POST /CertificateAuthority/AlertRecipients/CAThresholdRecipients method is used to create new recipients to receive CA threshold alerts in Keyfactor Command This method returns HTTP 200 OK on a success with the details of the threshold recipients.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Table 288: POST Certificate Authority Alert Recipients CA Threshold Recipients Input Body

Name	In	Description
Emails	Body	**Required**. An object containing a set of strings with the email address of each recipient. For example: `{ "Emails": [ "Recipient1@keyexample.com","Recipient3@keyexample.com" ] }`

Table 289: POST Certificate Authority Alert Recipients CA Threshold Recipients Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the recipient.
Email	A string indicating the email address of the recipient. **Note:** Only recipients created with the POST request are returned in the response. Any pre-existing recipients are not included in the response.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.20 PUT Certificate Authority Alert Recipients CA Threshold Recipients ID

The PUT /CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id} method is used to update the CA threshold alert recipient with the specified ID. This method returns HTTP 200 OK on a success with the details submitted.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Table 290: PUT Certificate Authority Alert Recipients CA Threshold Recipients {id} Input Body

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the CA threshold recipient. Use the *GET /CertificateAuthority/AlertRecipients/CAThresholdRecipients* method (see GET Certificate Authority Alert Recipients CA Threshold Recipients on page 1291) to retrieve the ID.
Email	Body	**Required**. A string indicating the updated email address of the recipient.

Table 291: PUT Certificate Authority Alert Recipients CA Threshold Recipients {id} Response Body

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the CA threshold recipient.
Email	A string indicating the new email address of the CA threshold recipient.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.8.21 POST Certificate Authority Import

The POST /CertificateAuthority/Import method is used to import into Keyfactor Command any certificate authorities from the provided configuration tenant DNS suffix (e.g. keyexample.com). This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_authorities/modify/

Table 292: POST Certificate Authority Import Input Parameters

Name	In	Description
dns	Query	**Required**. The DNS suffix of the configuration tenant to query for CAs to import. For example:

Name	In	Description
		`https://keyfactor.keyexample.com/KeyfactorAPI/CertificateAuthority/Import?dns=keyother.com`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9 Certificate Collections

The Certificate Collections component of the Keyfactor API is used to create, edit, list, and set permissions on certificate collections.

Table 293: Certificate Collections Endpoints

Endpoint	Method	Description	Link
/	GET	Returns all certificate collections with details about the collection configuration.	GET Certificate Collections on the next page
/	POST	Creates a new certificate collection.	POST Certificate Collections on page 1299
/	PUT	Updates an existing certificate collection.	PUT Certificate Collections on page 1305
/{id}	GET	Returns the certificate collection with the specified ID.	GET Certificate Collections ID on page 1309
/{id}	DELETE	Deletes the certificate collection with the specified ID.	DELETE Certificate Collection ID on page 1311
/{name}	GET	Returns the certificate collection with the specified name.	GET Certificate Collections Name on page 1311
/Copy	POST	Creates a new certificate collection	POST Certificate

Endpoint	Method	Description	Link
		based on an existing collection.	Collections Copy on page 1314
/NavItems	GET	Returns the list of *favorite* certificate collections (that have been set to *Show in Navigator*).	GET Certificate Collection Nav Items on page 1320
/{id}/Favorite	PUT	Updates the *favorite* setting for the collection specified.	PUT Certificate Collection ID Favorite on page 1320
/CollectionList	GET	Returns information about the definitions for all collections, including the de-duplication setting.	GET Certificate Collections List on page 1321

3.6.9.1 GET Certificate Collections

The GET /CertificateCollections method is used to return a list of all certificate collections. This method returns HTTP 200 OK on a success with details about each defined certificate collection. This method allows URL parameters to specify paging and the level of information detail.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 294: GET Certificate Collections Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • Name • Query • Favorite
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 295: GET Certificate Collections Response Data

Name	Description
ID	The Keyfactor Command reference ID for the certificate collection. The ID is automatically assigned by Keyfactor Command.
Name	The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.
Description	The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
Automated	An internally used Keyfactor Command field.
Content	A string containing the search criteria for the collection.
DuplicationField	An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. Possible values are:
ShowOnDashboard	A Boolean that sets whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false).
Favorite	A Boolean that sets whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false).

Value	Description
0	None
1	Common Name
2	Distinguished Name
3	Principal Name
4	Keyfactor Renewal

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.2 POST Certificate Collections

The POST /CertificateCollections method is used to create a new saved collection of certificates or update an existing collection. This method returns HTTP 200 OK on a success with details about the certificate collection.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
/certificates/collections/modify/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
/certificates/collections/modify/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 296: POST Certificate Collections Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Name</td><td>Body</td><td>Required. The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.</td></tr>
<tr><td>Description</td><td>Body</td><td>Required. The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
See also CopyFromId.</td></tr>
<tr><td>Query</td><td>Body</td><td>Required. A string containing the search criteria for the collection.
For example:
<pre>"Query": "(IssuedDate -ge \"%TODAY-7%\" AND TemplateShortName -ne NULL) OR (IssuedDate -ge \"%TODAY-7%\" AND IssuerDN -contains \"keyexample\")"</pre>
See Certificate Search Page on page 34 for querying guidelines.
See also CopyFromId.</td></tr>
<tr><td>DuplicationField</td><td>Body</td><td>An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. The default is 0. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>None</td></tr><tr><td>1</td><td>Common Name</td></tr><tr><td>2</td><td>Distinguished Name</td></tr><tr><td>3</td><td>Principal Name</td></tr><tr><td>4</td><td>Keyfactor Renewal</td></tr></table></td></tr>
</table>

Name	In	Description
ShowOnDashboard	Body	A Boolean that sets whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false). The default is *false*.
Favorite	Body	A Boolean that sets whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false). The default is *false*.
CopyFromId	Body	An integer identifying an existing certificate collection from which to copy the query string. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 1296) to locate the ID of the collection whose query you wish to copy. When you use this parameter, the permissions, query and description of the existing collection are copied to the new collection. Providing the *Query* or *Description* parameter in the request overrides the copied value and replaces it with the value provided in the request if the requesting user has global *Read* permissions for certificates. If the requesting user is granted *Read* permissions to the collection via collection-level security rather than global security, the *Query* the user provides will be appended to the existing query rather than overwriting it. See the below example. **Example:** Gina wants to create a new collection using the *CopyFromId* option. She first uses GET /CertificateCollections/{id} to list the collection she plans to copy from and sees the following results: `{ "Id": 10, "Name": "Keyexample Collection", "Description": "Certificates in the Keyexample Domain", "Automated": false, "Content": "CN -contains \"keyexample.com\"", "DuplicationField": 2, "ShowOnDashboard": false, "Favorite": true }` Gina wants her new certificate collection to retain the same collection-level permissions as the *Keyexample Collection*. However, she wants the collection to report on a different domain name. The *Keyexample Collection* is configured to grant collection-level permissions of *Read*, *Edit Metadata*,

Name	In	Description

		and *Download with Private Key* to the *Power Users* role.
		At the Key Example company, users with the Power Users role do not have global certificate *Read* permissions because all certificate permissions are granted using certificate collection permissions. Only full Keyfactor Command administrators have global certificate *Read* permissions. Users with the Power Users role have *Modify* permissions for certificate collections to allow them to create new collections. This level of permissions is significant for what Gina wants to do. Gina holds the Power Users role and is not a full administrator.
		Gina uses POST /CertificateCollections/Copy (or POST /CertificateCollections—the behavior and output would be the same) to create a new certificate collection using the *CopyFromId* option with the following command:
		`{ "CopyFromId": 10, "Name": "Keyother Collection", "Description": "Certificates in the Keyother Domain", "Query": "CN -contains \"keyother.com\"", "DuplicationField": 2, "ShowOnDashboard": false, "Favorite": true }`
		In the response, Gina sees the following:
		`{ "Id": 15, "Name": "Keyother Collection", "Description": "Certificates in the Keyother Domain", "Automated": false, "Content": "(CN -contains \"keyexample.com\") AND (CN -contains \"keyother.com\")", "Query": "(CN -contains \"keyexample.com\") AND (CN -contains \"keyother.com\")", "DuplicationField": 2, "ShowOnDashboard": false, "Favorite": true }`
		Notice that Gina has not achieved her desired goal. The

Name	In	Description

		new collection contains a query for both the keyexample.com domain and the keyother.com domain. Gina's new query was appended to the existing query rather than overwriting the existing query. This happened because Gina does not have global *Read* permissions for certificates and is done to prevent a user from increasing the scope of certificates they can view.
		Gina asks Martha, who is a full Keyfactor Command administrator and has the global *Read* permissions for certificates, to copy the collection for her. Martha first deletes the first Keyother Collection that Gina created and then runs the same command that Gina ran to create a new collection.
		In the response, Martha sees the following:
		<pre>{
		"Id": 16,
		"Name": "Keyother Collection",
		"Description": "Certificates in the Keyother Domain",
		"Automated": false,
		"Content": "(CN -contains \"keyother.com\")",
		"Query": "(CN -contains \"keyother.com\")",
		"DuplicationField": 2,
		"ShowOnDashboard": false,
		"Favorite": true
		}</pre>
		Notice that when Martha runs the command, Gina's goal is achieved.

Table 297: POST Certificate Collections Response Data

Name	Description
ID	The Keyfactor Command reference ID for the certificate collection. The ID is automatically assigned by Keyfactor Command.
Name	The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.
Description	The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
Content	A string containing the search criteria for the collection. This field contains the same value as *Query* and is retained for backwards compatibility.
Query	A string containing the search criteria for the collection.
DuplicationField	An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. Possible values are:
ShowOnDashboard	A Boolean that sets whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false).
Favorite	A Boolean that sets whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false).

Value	Description
0	None
1	Common Name
2	Distinguished Name
3	Principal Name
4	Keyfactor Renewal

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.3 PUT Certificate Collections

The PUT /CertificateCollections method is used to update an existing saved collection of certificates. This method returns HTTP 200 OK on a success with details about the certificate collection.

Note: Certificate collections that are configured for *Certificate Entered Collection* or *Certificate Left Collection* workflows (see Workflow Definition Operations on page 235) cannot be edited. This is done to prevent triggering a large number of entered/left workflows.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
/certificates/collections/modify/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
/certificates/collections/modify/#/ (where # is a reference to a specific certificate collection ID)

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 298: PUT Certificate Collections Input Parameters

Name	In	Description
ID	Body	**Required**. The Keyfactor Command reference ID for the certificate collection. The ID is automatically assigned by Keyfactor Command. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 1296) to locate the ID of the collection you wish to update.
Name	Body	**Required**. The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.
Description	Body	**Required**. The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
Query	Body	**Required**. A string containing the search criteria for the collection. For example: `"Query": "(IssuedDate -ge \"%TODAY-7%\" AND TemplateShortName -ne NULL) OR (IssuedDate -ge \"%TODAY-7%\" AND IssuerDN -contains \"keyexample\")"` See Certificate Search Page on page 34 for querying guidelines.
DuplicationField	Body	An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. The default is 0. Possible values are: **Value** / **Description** 0 — None 1 — Common Name

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td><table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>2</td><td>Distinguished Name</td></tr>
<tr><td>3</td><td>Principal Name</td></tr>
<tr><td>4</td><td>Keyfactor Renewal</td></tr>
</table></td></tr>
<tr><td>ShowOnDashboard</td><td>Body</td><td>A Boolean that sets whether the results from this collection are included on the Management Portal dashboard Certificate Counts by Collection graph (true) or not (false). The default is false.</td></tr>
<tr><td>Favorite</td><td>Body</td><td>A Boolean that sets whether the collection appears on the Navigator—on the Certificates top-level menu dropdown—(true) or not (false). The default is false.</td></tr>
</table>

Table 299: PUT Certificate Collections Response Data

Name	Description
ID	The Keyfactor Command reference ID for the certificate collection. The ID is automatically assigned by Keyfactor Command.
Name	The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.
Description	The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
Content	A string containing the search criteria for the collection. This field contains the same value as *Query* and is retained for backwards compatibility.
Query	A string containing the search criteria for the collection.
DuplicationField	An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. Possible values are: Value / Description 0 / None 1 / Common Name 2 / Distinguished Name 3 / Principal Name 4 / Keyfactor Renewal
ShowOnDashboard	A Boolean that sets whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false).
Favorite	A Boolean that sets whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.4 GET Certificate Collections ID

The GET /CertificateCollections/{id} method is used to retrieve details for a certificate collection with the specified ID. This method returns HTTP 200 OK on a success with details for the certificate collection.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 300: GET Certificate Collections {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the ID of the certificate collection to retrieve. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 1296) to retrieve a list of all the certificate collections to determine the certificate collection ID.

Table 301: GET Certificate Collections {id} Response Data

Name	Description
ID	The Keyfactor Command reference ID for the certificate collection. The ID is automatically assigned by Keyfactor Command.
Name	The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.
Description	The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
Automated	An internally used Keyfactor Command field.
Content	A string containing the search criteria for the collection.
DuplicationField	An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. Possible values are:
ShowOnDashboard	A Boolean that sets whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false).
Favorite	A Boolean that sets whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false).

Value	Description
0	None
1	Common Name
2	Distinguished Name
3	Principal Name
4	Keyfactor Renewal

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.5 DELETE Certificate Collection ID

The DELETE /CertificateCollections/{id} method is used to delete the certificate collection with the specified ID from the Keyfactor Command database. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/modify/
OR
/certificates/collections/modify/#/ (where # is a reference to a specific certificate collection ID)

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 302: DELETE Certificate Collection {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the certificate collection to delete. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 1296) to retrieve a list of certificate collections to determine the collection ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.6 GET Certificate Collections Name

The GET /CertificateCollections/{name} method is used to retrieve details for a certificate collection with the specified name. This method returns HTTP 200 OK on a success with details for the certificate collection.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 303: GET Certificate Collections Name Input Parameters

Name	In	Description
name	Path	**Required**. A string indicating the name of the certificate collection to retrieve. Use the *GET /CertificateCollections* method (see GET Certificates on page 1141) to retrieve a list of all the certificate collections to determine the certificate collection name. **Tip:** When using the Keyfactor API Reference and Utility, provide this name without quotation marks.

Table 304: GET Certificate Collections ID Response Data

Name	Description
ID	The Keyfactor Command reference ID for the certificate collection. The ID is automatically assigned by Keyfactor Command.
Name	The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.
Description	The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
Automated	An internally used Keyfactor Command field.
Content	A string containing the search criteria for the collection.
DuplicationField	An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. Possible values are:
ShowOnDashboard	A Boolean that sets whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false).
Favorite	A Boolean that sets whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false).

Value	Description
0	None
1	Common Name
2	Distinguished Name
3	Principal Name
4	Keyfactor Renewal

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.7 POST Certificate Collections Copy

The POST /CertificateCollections/Copy method is used to copy an existing saved collection of certificates in order to create a new collection. The permissions, query and description of the existing collection are copied to the new collection. Providing the *Query* or *Description* parameter in the request overrides the copied value and replaces it with the value provided in the request. This method returns HTTP 200 OK on a success with details about the new certificate collection.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
/certificates/collections/modify/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
/certificates/collections/modify/#/ (where # is a reference to a specific certificate collection ID)
Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 305: POST Certificate Collections Copy Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Name</td><td>Body</td><td>Required. The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.</td></tr>
<tr><td>Description</td><td>Body</td><td>Required. The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
See also CopyFromId.</td></tr>
<tr><td>Query</td><td>Body</td><td>Required. A string containing the search criteria for the collection.
For example:
<pre>"Query": "(IssuedDate -ge \"%TODAY-7%\" AND TemplateShortName -ne NULL) OR (IssuedDate -ge \"%TODAY-7%\" AND IssuerDN -contains \"keyexample\")"</pre>
See Certificate Search Page on page 34 for querying guidelines.
See also CopyFromId.</td></tr>
<tr><td>DuplicationField</td><td>Body</td><td>An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. The default is 0. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>None</td></tr><tr><td>1</td><td>Common Name</td></tr><tr><td>2</td><td>Distinguished Name</td></tr><tr><td>3</td><td>Principal Name</td></tr><tr><td>4</td><td>Keyfactor Renewal</td></tr></table></td></tr>
</table>

Name	In	Description
ShowOnDashboard	Body	A Boolean that sets whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false). The default is *false*.
Favorite	Body	A Boolean that sets whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false). The default is *false*.
CopyFromId	Body	An integer identifying an existing certificate collection from which to copy the query string. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 1296) to locate the ID of the collection whose query you wish to copy. When you use this parameter, the permissions, query and description of the existing collection are copied to the new collection. Providing the *Query* or *Description* parameter in the request overrides the copied value and replaces it with the value provided in the request if the requesting user has global *Read* permissions for certificates. If the requesting user is granted *Read* permissions to the collection via collection-level security rather than global security, the *Query* the user provides will be appended to the existing query rather than overwriting it. See the below example. **Example:** Gina wants to create a new collection using the *CopyFromId* option. She first uses GET /CertificateCollections/{id} to list the collection she plans to copy from and sees the following results: `{ "Id": 10, "Name": "Keyexample Collection", "Description": "Certificates in the Keyexample Domain", "Automated": false, "Content": "CN -contains \"keyexample.com\"", "DuplicationField": 2, "ShowOnDashboard": false, "Favorite": true }` Gina wants her new certificate collection to retain the same collection-level permissions as the *Keyexample Collection*. However, she wants the collection to report on a different domain name. The *Keyexample Collection* is configured to grant collection-level permissions of *Read*, *Edit Metadata*,

Name	In	Description

		and *Download with Private Key* to the *Power Users* role.
		At the Key Example company, users with the Power Users role do not have global certificate *Read* permissions because all certificate permissions are granted using certificate collection permissions. Only full Keyfactor Command administrators have global certificate *Read* permissions. Users with the Power Users role have *Modify* permissions for certificate collections to allow them to create new collections. This level of permissions is significant for what Gina wants to do. Gina holds the Power Users role and is not a full administrator.
		Gina uses POST /CertificateCollections/Copy (or POST /CertificateCollections—the behavior and output would be the same) to create a new certificate collection using the *CopyFromId* option with the following command:
		`{ "CopyFromId": 10, "Name": "Keyother Collection", "Description": "Certificates in the Keyother Domain", "Query": "CN -contains \"keyother.com\"", "DuplicationField": 2, "ShowOnDashboard": false, "Favorite": true }`
		In the response, Gina sees the following:
		`{ "Id": 15, "Name": "Keyother Collection", "Description": "Certificates in the Keyother Domain", "Automated": false, "Content": "(CN -contains \"keyexample.com\") AND (CN -contains \"keyother.com\")", "Query": "(CN -contains \"keyexample.com\") AND (CN -contains \"keyother.com\")", "DuplicationField": 2, "ShowOnDashboard": false, "Favorite": true }`
		Notice that Gina has not achieved her desired goal. The

Name	In	Description

		new collection contains a query for both the keyexample.com domain and the keyother.com domain. Gina's new query was appended to the existing query rather than overwriting the existing query. This happened because Gina does not have global *Read* permissions for certificates and is done to prevent a user from increasing the scope of certificates they can view.
		Gina asks Martha, who is a full Keyfactor Command administrator and has the global *Read* permissions for certificates, to copy the collection for her. Martha first deletes the first Keyother Collection that Gina created and then runs the same command that Gina ran to create a new collection.
		In the response, Martha sees the following:
		<pre>{
		"Id": 16,
		"Name": "Keyother Collection",
		"Description": "Certificates in the Keyother Domain",
		"Automated": false,
		"Content": "(CN -contains \"keyother.com\")",
		"Query": "(CN -contains \"keyother.com\")",
		"DuplicationField": 2,
		"ShowOnDashboard": false,
		"Favorite": true
		}</pre>
		Notice that when Martha runs the command, Gina's goal is achieved.

Table 306: POST Certificate Collections Copy Response Data

Name	Description
ID	The Keyfactor Command reference ID for the certificate collection. The ID is automatically assigned by Keyfactor Command.
Name	The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.
Description	The description for the collection. This description appears at the top of the page in the Management Portal for this collection and can be more detailed than the collection name.
Content	A string containing the search criteria for the collection. This field contains the same value as *Query* and is retained for backwards compatibility.
Query	A string containing the search criteria for the collection.
DuplicationField	An integer that sets the type of de-duplication (a.k.a. ignore renewed certificate results by) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. Possible values are:
ShowOnDashboard	A Boolean that sets whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false).
Favorite	A Boolean that sets whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false).

Value	Description
0	None
1	Common Name
2	Distinguished Name
3	Principal Name
4	Keyfactor Renewal

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.8 GET Certificate Collection Nav Items

The GET /CertificateCollection/NavItems method is used to return a list of the collections that have been set as favorites to *Show in Navigator*. This method returns HTTP 200 OK on a success with the collection names and IDs. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)

Table 307: GET Certificate Collection Nav Items Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the certificate collection.
Name	A string indicating the name of the certificate collection.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.9 PUT Certificate Collection ID Favorite

The PUT /CertificateCollection/{id}/Favorite method is used to update the *Favorite* / *Show in Navigator* setting for a collection. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/modify/

OR
/certificates/collections/modify/#/ (where # is a reference to a specific certificate collection ID)

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 308: PUT Certificate Collection{id} Favorite Input Body

Name	In	Description
id	Path	An integer indicating the Keyfactor Command reference ID of the certificate collection. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 1296) to retrieve a list of certificate collections to determine the collection ID.
ShowInNavigator	Body	A Boolean indicating whether the certificate collection should appear on the menu (true) or not(false). For example: `{ "ShowInNavigator": true }`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.9.10 GET Certificate Collections List

The GET /CertificateCollections/CollectionList method is used to return the definitions for all certificate collections. This method returns HTTP 200 OK on a success with details for each certificate collection, including the de-duplication setting.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Table 309: GET Certificate Collections List Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Collection Manager Search Feature on page 89. The query fields supported for this endpoint are: • Name • Query
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 310: GET Certificate Collections List Response Data

Name	Description
ID	The Keyfactor Command reference ID for the certificate collection. The ID is automatically assigned by Keyfactor Command.
Name	The name for the certificate collection. This name appears at the top of the page in the Keyfactor Command Management Portal for this collection and can be configured to appear on the Management Portal menu under Certificate Collections. It will also appear in other places within the Management Portal where you can reference certificate collections (e.g. expiration alerts and certain reports). Because it can appear on the menu and in selection dropdowns, the name should be fairly short.
Content	A string containing the search criteria for the collection. This field contains the same value as *Query* and is retained for backwards compatibility.
DuplicationField	A string indicating the type of de-duplication (a.k.a. *ignore renewed certificate results by*) to apply to the collection when using the collection in areas of Keyfactor Command that apply de-duplication (e.g. expiration alerts). For more information, see Saving Search Criteria as a Collection on page 42. Possible values are:

Value	Description
0	None
1	Common Name
2	Distinguished Name
3	Principal Name
4	Keyfactor Renewal

Favorite	A Boolean that indicates whether the collection appears on the Navigator—on the *Certificates* top-level menu dropdown—(true) or not (false).
ShowOnDashboard	A Boolean that indicates whether the results from this collection are included on the Management Portal dashboard *Certificate Counts by Collection* graph (true) or not (false).
HasQueryPermissions	A Boolean that indicates whether the user has query permissions (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10 Certificate Stores

The CertificateStores component of the Keyfactor API (formerly known as the JKS API) provides a set of methods to support management of certificate locations.

Through different remote Keyfactor orchestrators, Keyfactor Command can inventory, install, and remove certificates for each of the store types. For certain store types, additional actions are supported as well. The CertificateStores component provides a way to programmatically schedule jobs for these stores. For more information about certificate stores and their support within Keyfactor Command, see the *Keyfactor Command Reference Guide* and *Keyfactor Command Orchestrator Installation and Configuration Guide*, or contact your Keyfactor representative. The set of methods in this API component that can be used to manage certificate stores and their scheduled jobs is listed in Table 311: Certificate Stores Endpoints.

Table 311: Certificate Stores Endpoints

Endpoint	Method	Description	
/	DELETE	Deletes multiple certificate stores specified in the request body.	DELETE Certificate Stores on page 1326
/	GET	Returns all certificate stores with paging and option to specify detail level.	GET Certificate Stores on page 1327
/	POST	Creates a new certificate store if valid parameters are supplied.	POST Certificate Stores on page 1339
/	PUT	Updates an existing certificate store.	PUT Certificate Stores on page 1367
/{id}	DELETE	Deletes a certificate store by its GUID.	DELETE Certificate Stores ID on page 1397
/{id}	GET	Returns certificate store details for the specified certificate store.	GET Certificate Stores ID on page 1398
/{id}/Inventory	GET	Returns certificate inventory for the specified certificate store.	GET Certificate Stores ID Inventory

Endpoint	Method	Description	
			on page 1416
/Server (*deprecated)	GET	Returns a list of certificate store servers.	GET Certificate Stores Server on page 1418
/Server (*deprecated)	POST	Creates a new certificate store server.	POST Certificate Stores Server on page 1421
/Server (*deprecated)	PUT	Updates an existing certificate store server.	PUT Certificate Stores Server on page 1426
/Password	PUT	Updates the password for a certificate store.	PUT Certificate Stores Password on page 1431
/DiscoveryJob	PUT	Creates a job to find certificate stores.	PUT Certificate Stores Discovery Job on page 1434
/AssignContainer	PUT	Assigns a certificate store to a container.	PUT Certificate Stores Assign Container on page 1440
/Approve	POST	Approves an array of pending certificate stores.	POST Certificate Stores Approve on page 1451
/Schedule	POST	Creates an inventory schedule for a certificate store.	POST Certificate Stores Schedule on page 1462
/Reenrollment	POST	Schedules a reenrollment of a certificate into a certificate store.	POST Certificate Stores Reenrollment on page 1465
/Certificates/Add	POST	Configures a management job to add a certificate to one or more stores with the provided schedule.	POST Certificate Stores Certificates Add on page 1467
/Certificates/Remove	POST	Configures a management job to remove a certificate from one or	POST Certificate Stores Certificates

Endpoint	Method	Description	
		more stores with the provided schedule.	Remove on page 1473

3.6.10.1 DELETE Certificate Stores

The DELETE /CertificateStores method is used to delete multiple certificate stores in one request. The certificate store GUIDs should be supplied in the request body as a JSON array of strings. This endpoint returns 204 with no content upon success. GUIDs of any certificate stores that could not be deleted are returned in the response body. Delete operations will continue until the entire array of GUIDs has been processed.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 312: DELETE Certificate Stores Input Parameters

Name	In	Description
IDs	Body	**Required**. An array of strings indicating Keyfactor Command certificate store GUIDs for certificate stores that should be deleted in the form: `[52fe526d-9914-4239-b74b-b47d0607cf7c,8ec160d9-3242-4eb4-956b-a7651af6c542]` Use the *GET /CertificateStores* method (see GET Certificate Stores on the next page) to retrieve a list of all the certificate stores to determine the certificate store GUIDs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.2 GET Certificate Stores

The GET /CertificateStores method is used to return a list of all certificate stores defined in Keyfactor Command. The results include both approved certificates stores and certificates stores found on discovery but not yet approved. This method allows URL parameters to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details about the certificate store(s).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/
OR
/certificate_stores/read/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 313: GET Certificate Stores Input Parameters

Name	In	Description
queryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Certificate Store Search Feature on page 410. The query fields supported for this endpoint are: • AddSupported (True, False) • AgentAvailable (True, False) • AgentId • Approved (True, False) • Category (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol) • CertificateId • ClientMachine • Container (ContainerName) • ContainerId • HasInventoryScheduled (True, False) • PrivateKeyAllowed (0-Forbidden, 1-Optional, 2-Required) • RemoveSupported (True, False) • StorePath **Tip:** Use the following query to limit the results to only active certificate stores and not include discovery results: `approved -eq true`
pageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
returnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
sortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *ClientMachine*.

Name	In	Description
sortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 314: GET Certificate Stores Response Data

Name	Description
Id	A string indicating the GUID of the certificate store within Keyfactor Command. This ID is automatically set by Keyfactor Command.
ContainerId	An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Storepath	A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information. The maximum number of characters supported in this field is 722.
CertStoreInventoryJobId	A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
Approved	A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.
CreateIfMissing	A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.
Properties	Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information). When reading this field, the values are returned as simple key value pairs, with the

Name	Description

	values being individual values. When writing, the values are specified as objects, though they are typically single values.
	For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"`
	However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:
	`"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}}, \"ServerPassword\":{\"value\": {\"SecretValue\":\"MySuperSecretPassword\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	Note: There are three standard properties that are used for certificate store

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>types that require server credentials (e.g. F5):
ServerUsername
ServerPassword
ServerUseSsl
These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator for this store.</td></tr>
<tr><td>AgentAssigned</td><td>A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false).</td></tr>
<tr><td>ContainerName</td><td>A string indicating the name of the certificate store's associated container, if applicable.</td></tr>
<tr><td>InventorySchedule</td><td>An object indicating the inventory schedule for this certificate store. The following schedule types are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Tip: In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *ExactlyOnce*.

</td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

</td></tr>
<tr><td>ReenrollmentStatus</td><td>An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. The following reenrollment fields are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Data</td><td>A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.</td></tr>
<tr><td>Message</td><td>A string indicating the reason the certificate store cannot re-enroll, if applicable.</td></tr>
<tr><td>JobProperties</td><td>An array of strings containing the unique entry parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific entry parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is virtualServerName and is returned by GET CertificateStoreTypes like so:
<pre>"JobProperties": ["sniCert",
"virtualServerName"]</pre></td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>It can be seen in the Keyfactor Command Management Portal when editing the certificate store type on the *Entry Parameters* tab. The setting is referenced using the following format: <pre>"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]</pre> This field is optional.</td></tr><tr><td>CustomAliasAllowed</td><td>An integer indicating the option for a custom alias for this certificate store. • 0—forbidden • 1—optional • 2—required</td></tr><tr><td>EntryParameters</td><td>An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include: <table><tr><th>Name</th><th>Description</th></tr><tr><td>StoreTypeID</td><td>**Required**. An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above. If you are updating a certificate store type, this field is **required**.</td></tr><tr><td>Name</td><td>**Required**. A string containing the short name of the entry parameter. If you choose to define an entry parameter, this field is **required**. The name should be entered without spaces.</td></tr><tr><td>DisplayName</td><td>**Required**. A string containing the full display name of the entry para-</td></tr></table></td></tr></table>

Name	Description
	Name / **Description** Name / Description meter. If you choose to define an entry parameter, this field is **required**. Type: **Required**. A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret If you choose to define an entry parameter, this field is **required**. RequiredWhen: An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. The default is *false*. • **OnRemove**: If set to *true*, a value must be provided for this

Name	Description
	Name / **Description**
	Name / Description
	field when configuring a remove certificate job. The default is *false*. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*.
	DependsOn / A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.
	DefaultValue / A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*. This value is unset by default.
	Options / A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if *Type* is set to *MultipleChoice*. This value is unset by default.
	For example, to set a multiple choice entry parameter:

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><pre>"EntryParameter": [
	   {
	      "StoreTypeId": 111,
	      "Name": "ZooAnimal",
	      "DisplayName": "Favorite Zoo Animal",
	      "Type": "MultipleChoice",
	      "RequiredWhen": {
	         "HasPrivateKey": false,
	         "OnAdd": true,
	         "OnRemove": true,
	         "OnReenrollment": true
	      },
	      "DefaultValue": "Penguin",
	      "Options": "Tiger-
	,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
	   }
	]</pre>
	This value is unset by default.
	
	  **Tip:** What's the difference between properties (custom fields) and entry parameters?
	• Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record.
	• Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that</td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).</td></tr>
</table>
</td></tr>
<tr><td>SetNewPasswordAllowed</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false).</td></tr>
<tr><td>Password</td><td>Note: Secret data is stored in the secrets table or a PAM provider and is not returned in responses.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.3 POST Certificate Stores

The POST /CertificateStores method is used to create new certificate stores in Keyfactor Command. This method returns HTTP 200 OK on a success with details about the certificate store created.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 315: POST Certificate Stores Input Parameters

Name	In	Description
ContainerId	Body	An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).
ClientMachine	Body	**Required**. A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Storepath	Body	**Required**. A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information. The maximum number of characters supported in this field is 722.
CertStoreInventoryJobId	Body	A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.
CertStoreType	Body	**Required**. An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
Approved	Body	A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here. The default for new stores created with this method is *true*.
CreateIfMissing	Body	A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality. The default is *false*.
Properties	Body	**Required**. Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information). When reading this field, the values are returned as simple key value pairs, with the values being individual values. When writing, the values are specified as

Name	In	Description

		objects, though they are typically single values.
		For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
		`"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"`
		However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:
		`"{ \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"`
		An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:
		`"{ \"ServerUsername\":{\"value\": {\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"MySuperSecretPassword\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
		An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):
		`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`

Name	In	Description
		Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5): • ServerUsername • ServerPassword • ServerUseSsl These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
AgentId	Body	**Required**. A string indicating the Keyfactor Command GUID of the orchestrator for this store.
AgentAssigned	Body	A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false). The default is *true*.
ContainerName	Body	A string indicating the name of the certificate store's associated container, if applicable.
InventorySchedule	Body	An object indicating the inventory schedule for this certificate store. The following schedule types are supported:

Name	Description
Off	Turn off a previously configured schedule.
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Interval	A dictionary that indicates a job scheduled to run every x

Name	Description

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
   "Time": "2023-11-27T11:45:00Z"
}
</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ReenrollmentStatus</td><td>Body</td><td>An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. The following reenrollment fields are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Data</td><td>A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.</td></tr>
<tr><td>Message</td><td>A string indicating the reason the certificate store cannot re-enroll, if applicable.</td></tr>
<tr><td>JobProperties</td><td>An array of strings containing the unique entry parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific entry parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is virtualServerName and is returned by GET CertificateStoreTypes like so:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

<pre>"JobProperties": ["sniCert", "virtualServerName"]</pre>

It can be seen in the Keyfactor Command Management Portal when editing the certificate store type on the Entry Parameters tab.
The setting is referenced using the following format:

<pre>"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]</pre>

This field is optional.</td></tr>
<tr><td>CustomAliasAllowed</td><td>An integer indicating the option for a custom alias for this certificate store.
0—forbidden
1—optional
2—required</td></tr>
<tr><td>EntryParameters</td><td>An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>Required. A string containing the short name of the entry parameter. If you choose to define an entry parameter, this field is required. The name should be entered without spaces.</td></tr>
<tr><td>DisplayName</td><td>Required. A string containing the full display name of the entry parameter. If you choose to define an entry parameter, this field is required.</td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / **Description** **Name**: Type **Description**: **Required**. A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret If you choose to define an entry parameter, this field is **required**. **Name**: RequiredWhen **Description**: An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. The default is *false*. • **OnRemove**: If set to *true*, a value must be provided for

Name	In	Description
		Name / **Description** Name / Description this field when configuring a remove certificate job. The default is *false*. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*. DependsOn: A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value. DefaultValue: A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*. This value is unset by default. Options: A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if *Type* is set to

Name	In	Description
		Name / **Description**

Name	Description
	MultipleChoice. This value is unset by default.

For example, to set a multiple choice entry parameter:

```
"EntryParameter": [
   {
      "Name": "ZooAnimal",
      "DisplayName": "Favorite Zoo Animal",
      "Type": "MultipleChoice",
      "RequiredWhen": {
         "HasPrivateKey": false,
         "OnAdd": true,
         "OnRemove": true,
         "OnReenrollment": true
      },
      "DefaultValue": "Penguin",
      "Options": "Tiger-
,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
   }
]
```

This value is unset by default.

Tip: What's the difference between properties (custom fields) and entry parameters?

- Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record.
- Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).</td></tr>
</table>
</td></tr>
<tr><td>SetNewPasswordAllowed</td><td>Body</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false). The default is false.</td></tr>
<tr><td>Password</td><td>Body</td><td>An object indicating the source for and details of the credential information Keyfactor Command will use to access the certificates in a specific certificate store (the store password). This is different from credential information Keyfactor Command uses to access a certificate store server as a whole. The former (this setting) is typically used for Java keystores; the latter is typically used for certificates stores on Citrix NetScaler and F5 devices and set at the server level, not the certificate store level (see POST Certificate Stores on page 1339).
Certificate stores that require credentials support up to three possible credential options:
Use no store password.
This option is supported for Java keystores that would normally require a password, but can be configured with the no password option (see Value, below).
Store the credential information in the Keyfactor secrets table.
A Keyfactor secret is a user-defined password that is encrypted and stored securely in the Keyfactor Command database.
Load the credential information from a PAM provider.
See Privileged Access Management (PAM) on page 742 and PAM Providers on page 1871 for more information.
The possible values are:</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string—submitted as an object—indicating a password to be stored as a Keyfactor secret.

Tip: To set the no password option on a store, submit the password with a null value. For example:
<pre>"Password": {
   "SecretValue": {null}
}</pre>To set the value to a string to be stored in the Keyfactor secrets table, include the password in quotes. For example:
<pre>"Password": {
   "SecretValue": "MyVerySecurePassword"
}</pre></td></tr>
<tr><td>SecretTypeGuid</td><td>A string indicating the Keyfactor Command reference GUID for the type of credentials. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the secret provider. If you are using a secret provider with an integer ID, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the secret provider. If you are using a secret provider with a GUID ID, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>ProviderType ParameterValues</td><td>An array of objects containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / **Description** **Name** / **Description** reference ID for the PAM provider type parameter. Value: A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides). InstanceId: An integer indicating the Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used. InstanceGuid: A string indicating the Keyfactor Command reference GUID for the PAM provider. If you are attaching to something with a GUID ID, this will be used. Provider: An object containing information about the provider. PAM provider details include: **Name** / **Description** Id: An integer indicating the Keyfactor Command reference ID for the PAM provider. Name: A string indicating the internal name for the PAM provider. Area: An integer indicating the area of Keyfactor Command the provider is used for. PAM providers

Name	In	Description
		Name / Description Name / Description Name / Description generally have a value of 1, indicating they are used for certificate stores. ProviderType: An array of objects containing details about the provider type for the provider, including: Name / Description Id: A string indicating the Keyfactor Command reference GUID for the provider type. Name: A string that indicates the name of the provider type. ProviderType: An array of parameters that the

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Params</td><td>provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records.
See below instance of ProviderTypeParam for details.</td></tr>
</table>
</td></tr>
<tr><td>ProviderType ParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParams. See the previous level of ProviderTypeParamValues for details.</td></tr>
<tr><td>SecuredAreaId</td><td>An integer indicating the Keyfactor Command refer-</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / Description Name / Description Name / Description ence ID for the certificate store container the PAM provider is associated with, if any. This is considered deprecated and may be removed in a future release. Remote: A Boolean indicating whether the *Remote Provider* checkbox is checked when adding a new PAM provider (true), or not (false). See PAM Provider Configuration in Keyfactor Command on page 749. ProviderTypeParam: An array of objects that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. PAM provider type parameters include: Name / Description Id: An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter. Name: A string indicating the

Name	In	Description
		Name / Description Name / Description Name / Description internal name for the PAM provider type parameter. DisplayName: A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of *false*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of *true*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider. DataType: An integer indicating the data type for the parameter. Possible values are: • 1 = String • 2 = Secret InstanceLevel: A Boolean that sets whether the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (true).

Name	In	Description
		Name / Description Name / Description Name / Description For an example, see GET PAM Providers on page 1898. ProviderType — An object containing details for the provider type. Name / Description Id — A string indicating the Keyfactor Command reference GUID for the PAM provider type parameter. Name — A string indicating the

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>internal name for the PAM provider type parameter.</td></tr><tr><td>ProviderTypeParams</td><td>Unused field</td></tr></table></td></tr></table></td></tr></table></td></tr><tr><td>ProviderId</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider.</td></tr><tr><td>IsManaged</td><td>A Boolean indicating whether the credentials for the store are managed by a PAM provider (true) or stored in the Keyfactor secrets table (false). This value is automatically set by Keyfactor Command.</td></tr></table>

Table 316: POST Certificate Stores Response Data

Name	Description
Id	A string indicating the GUID of the certificate store within Keyfactor Command. This ID is automatically set by Keyfactor Command.
ContainerId	An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Storepath	A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information. The maximum number of characters supported in this field is 722.
CertStoreInventoryJobId	A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
Approved	A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.
CreateIfMissing	A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.
Properties	Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information). When reading this field, the values are returned as simple key value pairs, with the

Name	Description

	values being individual values. When writing, the values are specified as objects, though they are typically single values.
	For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"`
	However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:
	`"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}}, \"ServerPassword\":{\"value\": {\"SecretValue\":\"MySuperSecretPassword\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	Note: There are three standard properties that are used for certificate store

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>types that require server credentials (e.g. F5):
ServerUsername
ServerPassword
ServerUseSsl
These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator for this store.</td></tr>
<tr><td>AgentAssigned</td><td>A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false).</td></tr>
<tr><td>ContainerName</td><td>A string indicating the name of the certificate store’s associated container, if applicable.</td></tr>
<tr><td>InventorySchedule</td><td>An object indicating the inventory schedule for this certificate store. The following schedule types are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ReenrollmentStatus</td><td>
An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. The following reenrollment fields are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Data</td><td>A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.</td></tr>
<tr><td>Message</td><td>A string indicating the reason the certificate store cannot re-enroll, if applicable.</td></tr>
<tr><td>JobProperties</td><td>An array of strings containing the unique entry parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific entry parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is virtualServerName and is returned by GET CertificateStoreTypes like so:
<pre><code>"JobProperties": ["sniCert",
"virtualServerName"]</code></pre>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>It can be seen in the Keyfactor Command Management Portal when editing the certificate store type on the *Entry Parameters* tab. The setting is referenced using the following format: <pre>"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]</pre> This field is optional.</td></tr><tr><td>CustomAliasAllowed</td><td>An integer indicating the option for a custom alias for this certificate store. • 0—forbidden • 1—optional • 2—required</td></tr><tr><td>EntryParameters</td><td>An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include: <table><tr><th>Name</th><th>Description</th></tr><tr><td>StoreTypeID</td><td>**Required**. An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above. If you are updating a certificate store type, this field is **required**.</td></tr><tr><td>Name</td><td>**Required**. A string containing the short name of the entry parameter. If you choose to define an entry parameter, this field is **required**. The name should be entered without spaces.</td></tr><tr><td>DisplayName</td><td>**Required**. A string containing the full display name of the entry para-</td></tr></table></td></tr></table>

Name	Description
	Name / **Description** Name / Description meter. If you choose to define an entry parameter, this field is **required**. Type: **Required**. A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret If you choose to define an entry parameter, this field is **required**. RequiredWhen: An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. The default is *false*. • **OnRemove**: If set to *true*, a value must be provided for this

Name	Description
	Name / **Description** For example, to set a multiple choice entry parameter:

Name	Description
	field when configuring a remove certificate job. The default is *false*. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*.
DependsOn	A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.
DefaultValue	A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*. This value is unset by default.
Options	A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if *Type* is set to *MultipleChoice*. This value is unset by default.

Name	Description
	Name / **Description**

Name

Description

```
"EntryParameter": [
   {
      "StoreTypeId": 111,
      "Name": "ZooAnimal",
      "DisplayName": "Favorite Zoo Animal",
      "Type": "MultipleChoice",
      "RequiredWhen": {
         "HasPrivateKey": false,
         "OnAdd": true,
         "OnRemove": true,
         "OnReenrollment": true
      },
      "DefaultValue": "Penguin",
      "Options": "Tiger-
,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
   }
]
```

This value is unset by default.

Tip: What's the difference between properties (custom fields) and entry parameters?

- Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record.
- Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).</td></tr>
</table>
</td></tr>
<tr><td>SetNewPasswordAllowed</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false).</td></tr>
<tr><td>Password</td><td>Note: Secret data is stored in the secrets table or a PAM provider and is not returned in responses.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.4 PUT Certificate Stores

The PUT /CertificateStores method is used to update an existing certificate store in Keyfactor Command. This method returns HTTP 200 OK on a success with a message body containing the certificate store.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should

⚠ first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 317: PUT Certificate Stores Input Parameters

Name	In	Description
Id	Body	A string indicating the GUID of the certificate store within Keyfactor Command. This ID is automatically set by Keyfactor Command.
ContainerId	Body	An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).
ClientMachine	Body	**Required**. A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Storepath	Body	**Required**. A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information. The maximum number of characters supported in this field is 722.
CertStoreInventoryJobId	Body	A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.
CertStoreType	Body	**Required**. An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
Approved	Body	A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here. The default for new stores created with this method is *true*.
CreateIfMissing	Body	A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality. The default is *false*.
Properties	Body	**Required**. Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types

Name	In	Description

		on page 1546 for more information).
		When reading this field, the values are returned as simple key value pairs, with the values being individual values. When writing, the values are specified as objects, though they are typically single values.
		For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
		`"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"`
		However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:
		`"{ \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"`
		An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:
		`"{ \"ServerUsername\":{\"value\": {\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"MySuperSecretPassword\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
		An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):
		`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}},`

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<pre> \"ServerUseSsl\":{\"value\":\"true\"}
}"</pre>
Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):

ServerUsername
ServerPassword
ServerUseSsl

These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
</td></tr>
<tr><td>AgentId</td><td>Body</td><td>Required. A string indicating the Keyfactor Command GUID of the orchestrator for this store.</td></tr>
<tr><td>AgentAssigned</td><td>Body</td><td>A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false). The default is true.</td></tr>
<tr><td>ContainerName</td><td>Body</td><td>A string indicating the name of the certificate store's associated container, if applicable.</td></tr>
<tr><td>InventorySchedule</td><td>Body</td><td>An object indicating the inventory schedule for this certificate store. The following schedule types are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
</table>
</td></tr>
</table>

Name	In	Description

Name	Description
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour: `"Interval": { "Minutes": 60 }`
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2023-11-25T23:30:00Z" }`

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, exactly once at 11:45 am:

<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>

Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ReenrollmentStatus</td><td>Body</td><td>An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. The following reenrollment fields are supported:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Data</td><td>A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Message</td><td>A string indicating the reason the certificate store cannot re-enroll, if applicable.</td></tr>
<tr><td>JobProperties</td><td>An array of strings containing the unique entry parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific entry parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is virtualServerName and is returned by GET CertificateStoreTypes like so:

<pre>"JobProperties": ["sniCert", "virtualServerName"]</pre>
It can be seen in the Keyfactor Command Management Portal when editing the certificate store type on the Entry Parameters tab.
The setting is referenced using the following format:

<pre>"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]</pre>
This field is optional.</td></tr>
<tr><td>CustomAliasAllowed</td><td>An integer indicating the option for a custom alias for this certificate store.
• 0—forbidden
• 1—optional
• 2—required</td></tr>
<tr><td>EntryParameters</td><td>An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / **Description** **Name**: StoreTypeID **Description**: **Required**. An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above. If you are updating a certificate store type, this field is **required**. **Name**: Name **Description**: **Required**. A string containing the short name of the entry parameter. If you choose to define an entry parameter, this field is **required**. The name should be entered without spaces. **Name**: DisplayName **Description**: **Required**. A string containing the full display name of the entry parameter. If you choose to define an entry parameter, this field is **required**. **Name**: Type **Description**: **Required**. A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret If you choose to define an entry parameter, this field is **required**. **Name**: RequiredWhen **Description**: An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to

Name	In	Description
		Name / **Description** Name / Description *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. The default is *false*. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. The default is *false*. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*. DependsOn A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure

Name	In	Description
		Name / **Description**

Name	Description
	one custom parameter to display only if another custom parameter contains a value.
DefaultValue	A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*. This value is unset by default.
Options	A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if *Type* is set to *MultipleChoice*. This value is unset by default.

For example, to set a multiple choice entry parameter:

```
"EntryParameter": [
   {
      "StoreTypeId": 111,
      "Name": "ZooAnimal",
      "DisplayName": "Favorite Zoo Animal",
      "Type": "MultipleChoice",
      "RequiredWhen": {
         "HasPrivateKey": false,
         "OnAdd": true,
         "OnRemove": true,
         "OnReenrollment": true
      },
```

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
      "DefaultValue": "Penguin",
      "Options": "Tiger-
,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
   }
]
</pre>
This value is unset by default.

Tip: What's the difference between properties (custom fields) and entry parameters?

Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record.
Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).

</td></tr>
</table>
</td></tr>
<tr><td>SetNewPasswordAllowed</td><td>Body</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false). The default is false.</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Password</td><td>Body</td><td>An object indicating the source for and details of the credential information Keyfactor Command will use to access the certificates in a specific certificate store (the store password). This is different from credential information Keyfactor Command uses to access a certificate store server as a whole. The former (this setting) is typically used for Java keystores; the latter is typically used for certificates stores on Citrix NetScaler and F5 devices and set at the server level, not the certificate store level (see POST Certificate Stores on page 1339).
Certificate stores that require credentials support up to three possible credential options:
Use no store password.
This option is supported for Java keystores that would normally require a password, but can be configured with the no password option (see Value, below).
Store the credential information in the Keyfactor secrets table.
A Keyfactor secret is a user-defined password that is encrypted and stored securely in the Keyfactor Command database.
Load the credential information from a PAM provider.
See Privileged Access Management (PAM) on page 742 and PAM Providers on page 1871 for more information.
The possible values are:<table><tr><th>Name</th><th>Description</th></tr><tr><td>SecretValue</td><td>A string—submitted as an object—indicating a password to be stored as a Keyfactor secret.
Tip: To set the no password option on a store, submit the password with a null value. For example:<pre>"Password": {
   "SecretValue": {null}
}</pre>To set the value to a string to be stored in the Keyfactor secrets table, include the password in quotes. For example:<pre>"Password": {
   "SecretValue": "MyVerySecurePassword"
}</pre></td></tr><tr><td>SecretTy-</td><td>A string indicating the Keyfactor Command reference GUID</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>peGuid</td><td>for the type of credentials. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the secret provider. If you are using a secret provider with an integer ID, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the secret provider. If you are using a secret provider with a GUID ID, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>ProviderType ParameterValues</td><td>An array of objects containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr>
<tr><td>Value</td><td>A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr>
<tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used.</td></tr>
<tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the PAM provider. If you are attaching to something with a GUID ID, this will be used.</td></tr>
<tr><td>Provider</td><td>An object containing information about the provider. PAM provider details include:</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / Description Name / Description Name / Description Id — An integer indicating the Keyfactor Command reference ID for the PAM provider. Name — A string indicating the internal name for the PAM provider. Area — An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indicating they are used for certificate stores. ProviderType — An array of objects containing details about the provider type for the provider, including: Name / Description Id — A string indicating the Keyfactor Command reference

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>GUID for the provider type.</td></tr><tr><td>Name</td><td>A string that indicates the name of the provider type.</td></tr><tr><td>Provider Type Params</td><td>An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. See below instance of Provider-</td></tr></table></td></tr></table></td></tr></table></td></tr></table>

Name	In	Description
		Name: Description Name: Description Name: Description Name: Description Name: Description Nam-e: Descrip-tion *TypeParam* for details. ProviderTypeParamValues: An array of objects containing the values for the provider types specified by ProviderTypeParams. See the previous level of *ProviderTypeParamValues* for details. SecuredAreaId: An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any. This is considered deprecated and may be removed in a future release. Remote: A Boolean indicating whether the *Remote Provider* checkbox is checked when adding a new PAM provider (true), or not (false). See PAM Provider Configuration in

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>Keyfactor Command on page 749.</td></tr></table></td></tr><tr><td>ProviderTypeParam</td><td>An array of objects that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. PAM provider type parameters include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of false, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of true, this name appears on the Server</td></tr></table></td></tr></table></td></tr></table>

Name	In	Description
		Name / Description Name / Description Name / Description dialog for the parameter when a user creates a new PAM provider. DataType: An integer indicating the data type for the parameter. Possible values are: 1 = String, 2 = Secret InstanceLevel: A Boolean that sets whether the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (true). For an example, see GET PAM Providers on page 1898. ProviderType: An object containing details for the provider type. Name / Description Id: A string indic-

Name	In	Description
		Name / Description Name / Description Name / Description Name / Description ating the Keyfactor Command reference GUID for the PAM provider type parameter. Name / A string indicating the internal name for the PAM provider type parameter. ProviderTypeParams / Unused field

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td>ProviderId</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider.</td></tr><tr><td>IsManaged</td><td>A Boolean indicating whether the credentials for the store are managed by a PAM provider (true) or stored in the Keyfactor secrets table (false). This value is automatically set by Keyfactor Command.</td></tr></table>

Table 318: PUT Certificate Stores Response Data

Name	Description
Id	A string indicating the GUID of the certificate store within Keyfactor Command. This ID is automatically set by Keyfactor Command.
ContainerId	An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Storepath	A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information. The maximum number of characters supported in this field is 722.
CertStoreInventoryJobId	A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
Approved	A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.
CreateIfMissing	A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.
Properties	Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information). When reading this field, the values are returned as simple key value pairs, with the

Name	Description

	values being individual values. When writing, the values are specified as objects, though they are typically single values.
	For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"`
	However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:
	`"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}}, \"ServerPassword\":{\"value\": {\"SecretValue\":\"MySuperSecretPassword\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	Note: There are three standard properties that are used for certificate store

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>types that require server credentials (e.g. F5):
ServerUsername
ServerPassword
ServerUseSsl
These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator for this store.</td></tr>
<tr><td>AgentAssigned</td><td>A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false).</td></tr>
<tr><td>ContainerName</td><td>A string indicating the name of the certificate store’s associated container, if applicable.</td></tr>
<tr><td>InventorySchedule</td><td>An object indicating the inventory schedule for this certificate store. The following schedule types are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Tip: In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *ExactlyOnce*.

</td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

</td></tr>
<tr><td>ReenrollmentStatus</td><td>An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. The following reenrollment fields are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Data</td><td>A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.</td></tr>
<tr><td>Message</td><td>A string indicating the reason the certificate store cannot re-enroll, if applicable.</td></tr>
<tr><td>JobProperties</td><td>An array of strings containing the unique entry parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific entry parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is virtualServerName and is returned by GET CertificateStoreTypes like so:
<pre>"JobProperties": ["sniCert",
"virtualServerName"]</pre></td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>It can be seen in the Keyfactor Command Management Portal when editing the certificate store type on the *Entry Parameters* tab. The setting is referenced using the following format: <pre>"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]</pre> This field is optional.</td></tr><tr><td>CustomAliasAllowed</td><td>An integer indicating the option for a custom alias for this certificate store. • 0—forbidden • 1—optional • 2—required</td></tr><tr><td>EntryParameters</td><td>An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include: <table><tr><th>Name</th><th>Description</th></tr><tr><td>StoreTypeID</td><td>**Required**. An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above. If you are updating a certificate store type, this field is **required**.</td></tr><tr><td>Name</td><td>**Required**. A string containing the short name of the entry parameter. If you choose to define an entry parameter, this field is **required**. The name should be entered without spaces.</td></tr><tr><td>DisplayName</td><td>**Required**. A string containing the full display name of the entry para-</td></tr></table></td></tr></table>

Name	Description
	Name / **Description**
	Name / Description
	meter. If you choose to define an entry parameter, this field is **required**.
	Type — **Required**. A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret If you choose to define an entry parameter, this field is **required**.
	RequiredWhen — An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. The default is *false*. • **OnRemove**: If set to *true*, a value must be provided for this

Name	Description
	Name / **Description**
	Name / Description
	field when configuring a remove certificate job. The default is *false*. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*.
	DependsOn: A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.
	DefaultValue: A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*. This value is unset by default.
	Options: A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if *Type* is set to *MultipleChoice*. This value is unset by default.
	For example, to set a multiple choice entry parameter:

Name	Description
	Name / **Description**

```
"EntryParameter": [
   {
      "StoreTypeId": 111,
      "Name": "ZooAnimal",
      "DisplayName": "Favorite Zoo Animal",
      "Type": "MultipleChoice",
      "RequiredWhen": {
         "HasPrivateKey": false,
         "OnAdd": true,
         "OnRemove": true,
         "OnReenrollment": true
      },
      "DefaultValue": "Penguin",
      "Options": "Tiger-
,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
   }
]
```

This value is unset by default.

Tip: What's the difference between properties (custom fields) and entry parameters?

- Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record.
- Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).</td></tr>
</table>
</td></tr>
<tr><td>SetNewPasswordAllowed</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false).</td></tr>
<tr><td>Password</td><td>Note: Secret data is stored in the secrets table or a PAM provider and is not returned in responses.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.5 DELETE Certificate Stores ID

The DELETE /CertificateStores/{id} method is used to delete an existing certificate store with the specified GUID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 319: DELETE Certificate Stores Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the certificate store to delete. Use the *GET /CertificateStores* method (see GET Certificate Stores on page 1327) to retrieve a list of all the certificate stores to determine the certificate store GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.6 GET Certificate Stores ID

The GET /CertificateStores/{id} method is used to return details for the certificate store with the specified ID. This method returns HTTP 200 OK on a success with a message body containing certificate store details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/
OR
/certificate_stores/read/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 320: GET Certificate Stores {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the certificate store within Keyfactor Command. Use the GET /CertificateStore method (see GET Certificate Stores on page 1327) to retrieve a list of all the certificate stores to determine the certificate store GUID.

Table 321: GET Certificate Stores {id} Response Data

Name	Description
Id	A string indicating the GUID of the certificate store within Keyfactor Command. This ID is automatically set by Keyfactor Command.
ContainerId	An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Storepath	A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information. The maximum number of characters supported in this field is 722.
CertStoreInventoryJobId	A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
Approved	A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.
CreateIfMissing	A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.
Properties	Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information). When reading this field, the values are returned as simple key value pairs, with the

Name	Description

	values being individual values. When writing, the values are specified as objects, though they are typically single values.
	For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"`
	However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:
	`"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"MySuperSecretPassword\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):
	• ServerUsername

Name	Description
	• ServerPassword • ServerUseSsl These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
AgentId	A string indicating the Keyfactor Command GUID of the orchestrator for this store.
AgentAssigned	A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false).
ContainerName	A string indicating the name of the certificate store's associated container, if applicable.
InventorySchedule	An object indicating the inventory schedule for this certificate store. The following schedule types are supported:

Name	Description
Off	Turn off a previously configured schedule.
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

Name	Description

	Name / **Description**
	
	`"Interval": { "Minutes": 60 }`
	
	Daily
	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
	
	Name / **Description**
	Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
	
	For example, daily at 11:30 pm:
	
	`"Daily": { "Time": "2023-11-25T23:30:00Z" }`
	
	ExactlyOnce
	A dictionary that indicates a job scheduled to run at the time specified with the parameter:
	
	Name / Description
	Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
	
	For example, exactly once at 11:45 am:
	
	`"ExactlyOnce": { "Time": "2023-11-27T11:45:00Z" }`
	
	Tip: In some instances, jobs initially scheduled as *Imme-*

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>diate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ReenrollmentStatus</td><td>An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. The following reenrollment fields are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Data</td><td>A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.</td></tr>
<tr><td>Message</td><td>A string indicating the reason the certificate store cannot re-enroll, if applicable.</td></tr>
<tr><td>JobProperties</td><td>An array of strings containing the unique entry parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific entry parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is virtualServerName and is returned by GET CertificateStoreTypes like so:
<pre>"JobProperties": ["sniCert", "virtualServerName"]</pre>It can be seen in the Keyfactor Command Management Portal</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>when editing the certificate store type on the *Entry Parameters* tab. The setting is referenced using the following format: <pre>"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]</pre> This field is optional.</td></tr><tr><td>CustomAliasAllowed</td><td>An integer indicating the option for a custom alias for this certificate store. • 0—forbidden • 1—optional • 2—required</td></tr><tr><td>EntryParameters</td><td>An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include: <table><tr><th>Name</th><th>Description</th></tr><tr><td>StoreTypeID</td><td>**Required**. An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above. If you are updating a certificate store type, this field is **required**.</td></tr><tr><td>Name</td><td>**Required**. A string containing the short name of the entry parameter. If you choose to define an entry parameter, this field is **required**. The name should be entered without spaces.</td></tr><tr><td>DisplayName</td><td>**Required**. A string containing the full display name of the entry parameter. If you choose to define an entry para-</td></tr></table></td></tr></table>

Name	Description
	Name / **Description**

Name	Description
	meter, this field is **required**.
Type	**Required**. A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret If you choose to define an entry parameter, this field is **required**.
RequiredWhen	An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. The default is *false*. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. The default is *false*.

Name	Description
	Name / **Description**

Name	Description
	Name / Description

Name	Description
	• **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*.
DependsOn	A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.
DefaultValue	A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*. This value is unset by default.
Options	A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if *Type* is set to *MultipleChoice*. This value is unset by default.

For example, to set a multiple choice entry parameter:

```
"EntryParameter": [
   {
```

Name	Description
	Name / **Description**

```
      "StoreTypeId": 111,
      "Name": "ZooAnimal",
      "DisplayName": "Favorite Zoo Animal",
      "Type": "MultipleChoice",
      "RequiredWhen": {
         "HasPrivateKey": false,
         "OnAdd": true,
         "OnRemove": true,
         "OnReenrollment": true
      },
      "DefaultValue": "Penguin",
      "Options": "Tiger-
,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
   }
]
```

This value is unset by default.

Tip: What's the difference between properties (custom fields) and entry parameters?

- Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record.
- Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).</td></tr>
</table>
</td></tr>
<tr><td>SetNewPasswordAllowed</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false).</td></tr>
<tr><td>Password</td><td>An object indicating the source for and details of the credential information Keyfactor Command will use to access the certificates in a specific certificate store (the store password). This is different from credential information Keyfactor Command uses to access a certificate store server as a whole. The former (this setting) is typically used for Java keystores; the latter is typically used for certificates stores on Citrix NetScaler and F5 devices and set at the server level, not the certificate store level (see POST Certificate Stores on page 1339).
Certificate stores that require credentials support up to three possible credential options:
Use no store password.
This option is supported for Java keystores that would normally require a password, but can be configured with the no password option (see Value, below).
Store the credential information in the Keyfactor secrets table.
A Keyfactor secret is a user-defined password that is encrypted and stored securely in the Keyfactor Command database.
Load the credential information from a PAM provider.
See Privileged Access Management (PAM) on page 742 and PAM Providers on page 1871 for more information.
The possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string—submitted as an object—indicating a password to be stored as a Keyfactor secret.
Tip: To set the no password option on a store, submit the password with a null value. For example:
<pre>"Password": {</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
        "SecretValue": {null}
    }
</pre>
To set the value to a string to be stored in the Keyfactor secrets table, include the password in quotes. For example:
<pre>
"Password": {
    "SecretValue": "MyVerySecurePassword"
}
</pre>
</td></tr>
<tr><td>SecretTypeGuid</td><td>A string indicating the Keyfactor Command reference GUID for the type of credentials. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the secret provider. If you are using a secret provider with an integer ID, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the secret provider. If you are using a secret provider with a GUID ID, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>ProviderType ParameterValues</td><td>An array of objects containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr>
<tr><td>Value</td><td>A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Name	Description
Instance-Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used.
Instance-Guid	A string indicating the Keyfactor Command reference GUID for the PAM provider. If you are attaching to something with a GUID ID, this will be used.
Provider	An object containing information about the provider. PAM provider details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider.
Name	A string indicating the internal name for the PAM provider.
Area	An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indicating they are used for certificate stores.
Provider-Type	An array of objects containing details about the provider type for the provider, including:

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string that indicates the name of the provider type.
ProviderTypeParams	An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. See below instance of *Provider-*

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>*TypeParam* for details.</td></tr></table></td></tr><tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParams. See the previous level of *ProviderTypeParamValues* for details.</td></tr><tr><td>SecuredAreaId</td><td>An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>Remote</td><td>A Boolean indicating whether the *Remote Provider* checkbox is checked when adding a new PAM provider (true), or not (false). See PAM Provider Configuration in Keyfactor Command on page 749.</td></tr></table></td></tr><tr><td>ProviderType</td><td>An array of objects that the provider type uses for data input in Keyfactor Command when</td></tr></table></td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Param</td><td>creating new PAM provider and certificate store records. PAM provider type parameters include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of false, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of true, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>Instance-</td><td>A Boolean that sets whether</td></tr></table></td></tr></table></td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Level</td><td>the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (true). For an example, see <a>GET PAM Providers on page 1898.</td></tr><tr><td>ProviderType</td><td>An object containing details for the provider type.<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string</td></tr></table></td></tr></table></td></tr></table></td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>ProviderTypeParams</td><td>Unused field</td></tr></table></td></tr></table></td></tr></table></td></tr><tr><td>ProviderId</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider.</td></tr><tr><td>IsManaged</td><td>A Boolean indicating whether the credentials for the store are managed by a PAM provider (true) or stored in the Keyfactor secrets table (false). This value is automatically set by Keyfactor Command.</td></tr></table> **Note:** Secret data is stored in the secrets table or a PAM provider and is not returned in responses.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.7 GET Certificate Stores ID Inventory

The GET /CertificateStores/{id}/Inventory method is used to return a list of all the certificates found in the selected certificate store based on an inventory done using Keyfactor Command an approved orchestrator. The results include both end entity certificates and chain certificates found in the store. This method allows URL parameters to specify paging and sorting. This method returns HTTP 200 OK on a success with details about the certificates in the store.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/
OR
/certificate_stores/read/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 322: GET Certificate Stores {id} Inventory Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the certificate store within Keyfactor Command.
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 323: GET Certificate Stores {id} Inventory Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string indicating the alias for the certificate in the certificate store. The format for this varies depending on the certificate store type and whether the Overwrite flag is selected. See PFX Enrollment on page 146 for more information.</td></tr>
<tr><td>Certificates</td><td>An array of objects indicating the certificates (end entity and chain) found in the certificate store. Certificate details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the certificate.</td></tr>
<tr><td>IssuedDN</td><td>A string indicating the distinguished name of the certificate.</td></tr>
<tr><td>SerialNumber</td><td>A string indicating the serial number of the certificate.</td></tr>
<tr><td>NotBefore</td><td>A string indicating the date, in UTC, on which the certificate was issued by the certificate authority.</td></tr>
<tr><td>NotAfter</td><td>A string indicating the date, in UTC, on which the certificate expires.</td></tr>
<tr><td>SigningAlgorithm</td><td>A string indicating the algorithm used to sign the certificate.</td></tr>
<tr><td>IssuerDN</td><td>A string indicating the distinguished name of the issuer.</td></tr>
<tr><td>Thumbprint</td><td>A string indicating the thumbprint of the certificate.</td></tr>
<tr><td>CertStoreInventoryItemId</td><td>An integer indicating the Keyfactor Command referenced ID of the certificate in the certificate store.</td></tr>
</table></td></tr>
<tr><td>CertStoreInventoryItemId</td><td>An integer indicating the Keyfactor Command reference ID of the certificate in the certificate store.</td></tr>
<tr><td>Parameters</td><td>An object containing the entry parameters associated with the certificate</td></tr>
</table>

Name	Description
	in the certificate store. Expected entry parameters will vary depending on the configuration of the certificate store type. See POST Certificate Store Types on page 1552 for more information about entry parameters.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.8 GET Certificate Stores Server

The GET /CertificateStores/Server method is used to retrieve all servers for certificate stores. Only select types of certificate stores have an associated server. These include F5, IIS, Citrix\NetScaler, and any other custom method you've defined to support this. This method returns HTTP 200 OK on a success with details for each server.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/
OR
/certificate_stores/read/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Note: This method has been deprecated and will be removed from the Keyfactor API in release 12. Certificate store server information is now found in the Properties field of the certificate store (see GET Certificate Stores on page 1327).

Table 324: GET Certificate Stores Server Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Certificate Store Search Feature on page 410. The query fields supported for this endpoint are: • Id • Name • ServerType
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 325: GET Certificate Stores Server Response Data

Name	Description
Id	The ID of the server.
Username	The username used to connect to the certificate store. **Note:** Secret data is stored in the secrets table or a PAM provider and is not returned in responses.
Password	The password used to connect to the certificate store. **Note:** Secret data is stored in the secrets table or a PAM provider and is not returned in responses.
UseSSL	A Boolean that indicates whether Keyfactor Command will use SSL to communicate with the server (true) or not (false).
ServerType	An integer indicating the type of server. Possible values include (plus any custom values): **Value** / **Description** 0 – F5 Web Server & F5 SSL Profiles (Deprecated) 1 – NetScaler (Deprecated) 2 – FTP (Deprecated) 3 – F5 Web Server REST 4 – F5 SSL Profiles REST 5 – F5 CA Bundles REST
Name	The host name of the server.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.9 POST Certificate Stores Server

The POST /CertificateStores/Server method is used to create a new server record for a certificate store in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the newly created server record.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. Creating new certificate store server records requires permissions at the global level. See Container Permissions on page 629 for more information about global vs container permissions.

Note: This method has been deprecated and will be removed from the Keyfactor API in a future release. This method is retained until that time for backwards compatibility. Continuing to use this endpoint with the latest Keyfactor Command functionality could cause serious data issues. Certificate store server information is now found in the Properties field of the certificate store (see POST Certificate Stores on page 1339).

Tip: If a certificate store that requires a server is missing a server definition within the store record, the certificate store server created with this method will be used. If no credentials are supplied in the request and no certificate store server exists, an error is returned and the request fails.

Table 326: POST Certificate Stores Server Input Parameters

Name	In	Description
Username	Body	

Name	Description
SecretValue	A string containing the username. This value only needs to be supplied if you're storing your username in the Keyfactor Command database.
Provider	An integer that identifies the PAM provider used to store the username. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use. See PAM Provider Configuration in Keyfactor Command on page 749 in the *Keyfactor Command Reference Guide* for more information. This value only needs to be supplied if you're storing your username using a PAM provider.

Parameters	The parameters required by your PAM provider, containing the information that identifies the location of the username in the PAM solution. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of the parameters used by your PAM provider. Only parameters where *InstanceLevel* is equal to *true* need to be supplied in the request.
	For example, for Delinea (formerly Thycotic), this might be:
	`"Username": { "Provider": 2, "Parameters": { "SecretId": 4 } },`
	For CyberArk, this might be:
	`"Username": { "Provider": 5, "Parameters": { "Folder": "Root", "Object": "F5Username" } },`

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Password</td><td>Body</td><td>Required. The password used to connect to the certificate store. Password parameters include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the password. This value only needs to be supplied if you're storing your password in the Keyfactor Command database.</td></tr>
<tr><td>Provider</td><td>An integer that identifies the PAM provider used to store the password. Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use.</td></tr>
<tr><td>Parameters</td><td>The parameters required by your PAM provider, containing the information that identifies the location of the password in the PAM solution. Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of the parameters used by your PAM provider. Only parameters where InstanceLevel is equal to true need to be supplied in the request.
For example, for Delinea, this might be:

<pre>"Password": {
   "Provider": 2,
      "Parameters": {
      "SecretId": 5
   }
},</pre>

For CyberArk, this might be:

<pre>"Password": {
   "Provider": 5,
   "Parameters": {
      "Folder": "Root",
      "Object": "F5Password"
   }
},</pre></td></tr>
</table></td></tr>
<tr><td>UseSSL</td><td>Body</td><td>A Boolean that indicates whether Keyfactor Command will use SSL to</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>communicate with the server (true) or not (false). The default is false.</td></tr>
<tr><td>ServerType</td><td>Body</td><td>An integer indicating the type of server. Possible values include (plus any custom values):
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>F5 Web Server & F5 SSL Profiles (Deprecated)</td></tr><tr><td>1</td><td>NetScaler (Deprecated)</td></tr><tr><td>2</td><td>FTP (Deprecated)</td></tr><tr><td>3</td><td>F5 Web Server REST</td></tr><tr><td>4</td><td>F5 SSL Profiles REST</td></tr><tr><td>5</td><td>F5 CA Bundles REST</td></tr></table>
Use the GET /CertificateStoreTypes method (see <a>GET Certificate Store Types on page 1546) to locate the server types for your custom certificate store types. The ServerRegistration value returned by that method maps to the ServerType.
The default is <code>0</code>.</td></tr>
<tr><td>Name</td><td>Body</td><td>Required. The host name of the server.</td></tr>
<tr><td>Container</td><td>Body</td><td>An integer that identifies the certificate store container into which the certificate store should be placed for organizational and management purposes.</td></tr>
</table>

Table 327: POST Certificate Stores Server Response Data

Name	Description
Id	The ID of the server.
UseSSL	A Boolean that indicates whether Keyfactor Command will use SSL to communicate with the server (true) or not (false).
ServerType	An integer indicating the type of server. Possible values include (plus any custom values): **Value** / **Description** 0 – F5 Web Server & F5 SSL Profiles (Deprecated) 1 – NetScaler (Deprecated) 2 – FTP (Deprecated) 3 – F5 Web Server REST 4 – F5 SSL Profiles REST 5 – F5 CA Bundles REST
Name	The host name of the server.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.10 PUT Certificate Stores Server

The PUT /CertificateStores/Server method is used to update the server record for a certificate store in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the server record.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)

Permissions for certificate stores can be set at either the global or certificate store container level. Updating certificate store server records requires permissions at the global level. See Container Permissions on page 629 for more information about global vs container permissions.

Note: This method has been deprecated and will be removed from the Keyfactor API in a future release. This method is retained until that time for backwards compatibility. Continuing to use this endpoint with the latest Keyfactor Command functionality could cause serious data issues such as, for instance, overwriting all certificate stores on the server. Certificate store server information is now found in the Properties field of the certificate store (see PUT Certificate Stores on page 1367. This endpoint has additional functionality, such as being able to set different credentials for different stores on the same server.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 328: PUT Certificate Stores Server Input Parameters

Name	In	Description
Id	Body	The ID of the server.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Username</td><td>Body</td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the username.
This value only needs to be supplied if you're storing your username in the Keyfactor Command database.</td></tr>
<tr><td>Provider</td><td>An integer that identifies the PAM provider used to store the username. Use the GET /PamProviders method (see GET PAM Providers on page 1898) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use. See PAM Provider Configuration in Keyfactor Command on page 749 in the Keyfactor Command Reference Guide for more information.
This value only needs to be supplied if you're storing your username using a PAM provider.</td></tr>
<tr><td>Parameters</td><td>The parameters required by your PAM provider, containing the information that identifies the location of the username in the PAM solution. Use the GET /PamProviders method (see GET PAM Providers on page 1898) to retrieve a list of the parameters used by your PAM provider. Only parameters where InstanceLevel is equal to true need to be supplied in the request.
For example, for Delinea (formerly Thycotic), this might be:
<pre>"Username": {
   "Provider": 2,
   "Parameters": {
      "SecretId": 4
   }
},</pre>
For CyberArk, this might be:
<pre>"Username": {
   "Provider": 5,
   "Parameters": {
      "Folder": "Root",
      "Object": "F5Username"
   }
},</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Password</td><td>Body</td><td>Required. The password used to connect to the certificate store. Password parameters include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the password. This value only needs to be supplied if you're storing your password in the Keyfactor Command database.</td></tr>
<tr><td>Provider</td><td>An integer that identifies the PAM provider used to store the password. Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use.</td></tr>
<tr><td>Parameters</td><td>The parameters required by your PAM provider, containing the information that identifies the location of the password in the PAM solution. Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of the parameters used by your PAM provider. Only parameters where InstanceLevel is equal to true need to be supplied in the request.
For example, for Delinea, this might be:

<pre>"Password": {
   "Provider": 2,
      "Parameters": {
      "SecretId": 5
   }
},</pre>
For CyberArk, this might be:

<pre>"Password": {
   "Provider": 5,
   "Parameters": {
      "Folder": "Root",
      "Object": "F5Password"
   }
},</pre></td></tr>
</table></td></tr>
<tr><td>UseSSL</td><td>Body</td><td>A Boolean that indicates whether Keyfactor Command will use SSL to communicate with the server (true) or not (false). The default is false.</td></tr>
</table>

Name	In	Description
Name	Body	**Required**. The host name of the server.
Container	Body	An integer that identifies the certificate store container into which the certificate store should be placed for organizational and management purposes.

Table 329: PUT Certificate Stores Server Response Data

Name	Description
Id	The ID of the server.
UseSSL	A Boolean that indicates whether Keyfactor Command will use SSL to communicate with the server (true) or not (false).
ServerType	An integer indicating the type of server. Possible values include (plus any custom values): **Value** / **Description** 0 – F5 Web Server & F5 SSL Profiles (Deprecated) 1 – NetScaler (Deprecated) 2 – FTP (Deprecated) 3 – F5 Web Server REST 4 – F5 SSL Profiles REST 5 – F5 CA Bundles REST
Name	The host name of the server.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.11 PUT Certificate Stores Password

The PUT /CertificateStores/Password method is used to update a password for a certificate store that supports this functionality. This updates the password stored in Keyfactor Command for the

certificate store but does not update the certificate store itself. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 330: PUT Certificate Stores Password Input Parameters

Name	Type	Description
CertStoreID	Body	**Required**. A string indicating the GUID of the certificate store. Use the *GET CertificateStores* method (see GET Certificate Stores on page 1327) to retrieve a list of all your certificate stores to determine the GUID of the store.
NewPassword	Body	**Required**. A object that sets the password used by Keyfactor Command to access the certificate store. It does not impact the certificate store itself, just Keyfactor Command's definition of it. Password settings include:

Name	Description
SecretValue	A string containing the password. This value only needs to be supplied if you're storing your password in the Keyfactor Command database.
Provider	An integer that identifies the PAM provider used to store the password. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use. This value only needs to be supplied if you're storing your password using a PAM provider.

Parameters	An object containing the parameters required by your PAM provider, including the information that identifies the location of the password in the PAM solution. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of the parameters used by your PAM provider. Only parameters where *InstanceLevel* is equal to *true* need to be supplied in the request.
	For example, for Delinea (formerly Thycotic), this might be:
	`"NewPassword": {`
	`"Provider": 2,`
	`"Parameters": {`

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
    "SecretId": 5
  }
},
</pre>
For CyberArk, this might be:
<pre>
"NewPassword": {
  "Provider": 5,
  "Parameters": {
    "Folder": "Root",
    "Object": "F5Password"
  }
},
</pre>
</td></tr>
</table>
For a password stored in the Keyfactor Command database, this might be:
<pre>
"NewPassword": {
   "SecretValue": "P@ssw0rd"
}
</pre>
</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.12 PUT Certificate Stores Discovery Job

The PUT /CertificateStores/DiscoveryJob method is used to schedule a discovery job for certificate stores. The certificate store discovery feature is used to scan machines and devices for existing certificates and certificate stores, which can then be configured for management in Keyfactor Command. Certificate store discovery is supported for:

- PEM and Java certificate stores discovered by the Keyfactor Java Agent. Only stores to which the service account running the Keyfactor Java Agent has at least read permissions will be returned on a discover job.

- PEM, Java, F5, F5 bundle and SSL certificates discovered by the Keyfactor Universal Orchestrator with an appropriate custom extension. For more information about the Keyfactor Universal Orchestrator and custom extensions, see Universal Orchestrator on page 2879.
- Any custom certificate store types configured to support this function.

This endpoint returns 204 with no content upon success. The method schedules the discovery job through the orchestrator. The results of the discovery job are determined separately (see POST Certificate Stores Approve on page 1451).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 331: PUT Certificate Stores Discovery Job Input Parameters

Name	In	Description
ClientMachine	Body	**Required**. A string indicating the name in Keyfactor Command of the client machine that will do the discovery. This is not necessarily the actual DNS name of the server; the orchestrator may have been installed using an alternative as a reference name.
AgentId	Body	**Required**. A string indicating the Keyfactor Command reference GUID of the orchestrator for this store.
Type	Body	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+. The default is 0 for a JKS discovery using the Keyfactor Java Agent.
JobExecutionTimestamp	Body	The date and time at which the discovery job should run. If no date is provided, the job will be scheduled to run immediately. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Dirs	Body	**Required**. A string containing the directory or directories to search during the discovery job. Multiple directories should be separated by commas. • **Java** For Java discovery, enter at a minimum either "/" for a Linux server or "c:\" for a Windows server. • **PEM** For PEM discovery, enter at a minimum either "/" for a Linux server or "c:\" for a Windows server. • **F5** For F5 discovery, enter "/".
IgnoredDirs	Body	A string containing the directories that should not be included in the search. Multiple directories should be separated by commas.
Extensions	Body	A string containing the file extensions for which to search. For

Name	In	Description
		example, search for files with the extension *jks* in order to exclude files with other extensions such as *txt*. Use *noext* to search file files without extensions. The dot should not be included when specifying extensions.
NamePatterns	Body	A string against which to compare the file names of certificate store files and return only those that contain the specified string (e.g. *myjks*).
SymLinks	Body	A Boolean that sets whether the job should follow symbolic links on Linux and UNIX operating systems and report both the actual location of a found certificate store file in addition to the symbolic link pointing to the file. This option is ignored on Windows.
Compatibility	Body	A Boolean that sets whether the job will run using the compatibility mode introduced in Java version 1.8 to locate both JKS and PKCS12 type files (true) or not (false). This option applies only to Java keystore discover jobs.

Name	In	Description
ServerUsername	Body	

Name	Description
SecretValue	A string containing the username. This value only needs to be supplied if you're storing your username in the Keyfactor Command database.
Provider	An integer that identifies the PAM provider used to store the username. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use. See PAM Provider Configuration in Keyfactor Command on page 749 for more information. This value only needs to be supplied if you're storing your username using a PAM provider.

Parameters	The parameters required by your PAM provider, containing the information that identifies the location of the username in the PAM solution. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of the parameters used by your PAM provider. Only parameters where *InstanceLevel* is equal to *true* need to be supplied in the request.
	For example, for Delinea (formerly Thycotic), this might be:
	`"ServerUsername": { "Provider": 2, "Parameters": { "SecretId": 4 } },`
	For CyberArk, this might be:
	`"ServerUsername": { "Provider": 5, "Parameters": { "Folder": "Root", "Object": "F5Username" } }`

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ServerPassword</td><td>Body</td><td>Required*. The password used to connect to the certificate store server. Password parameters include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the password. This value only needs to be supplied if you're storing your password in the Keyfactor Command database.</td></tr>
<tr><td>Provider</td><td>An integer that identifies the PAM provider used to store the password. Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use.</td></tr>
<tr><td>Parameters</td><td>The parameters required by your PAM provider, containing the information that identifies the location of the password in the PAM solution. Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of the parameters used by your PAM provider. Only parameters where InstanceLevel is equal to true need to be supplied in the request.

For example, for Delinea, this might be:

<pre>"Password": {
  "Provider": 2,
  "Parameters": {
    "SecretId": 5
  }
},</pre>

For CyberArk, this might be:

<pre>"Password": {
  "Provider": 5,</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre> "Parameters": {
    "Folder": "Root",
    "Object": "F5Password"
  }
},</pre></td></tr>
</table>

Note: Secret data is stored in the secrets table or a PAM provider and is not returned in responses.

This field is required only for select certificate store types that require authentication at the server level. These includeF5, Citrix/NetScaler, IIS, and any custom method you've defined to support this.
</td></tr>
<tr><td>ServerUseSsl</td><td>Body</td><td>A Boolean that indicates whether Keyfactor Command will use SSL to communicate with the certificate store server (true) or not (false). The default is false.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.13 PUT Certificate Stores Assign Container

The PUT /CertificateStores/AssignContainer method is used to assign one or more certificate stores to a container. This method returns HTTP 200 OK on a success with the certificate stores that were just assigned to a container.

If you are creating a new container and assigning stores to it in one action, you should include the following fields:

- NewContainerName
- NewContainerType
- KeystoreIds

If you are assigning stores to an already existing container, you should include the following fields:

- CertStoreContainerId
- KeystoreIds

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 332: PUT Certificate Stores Assign Container Input Parameters

Name	In	Description
CertStoreContainerId	Body	**Required***. An integer that identifies the container into which you want to place the certificate store or stores. One of the following is **required**: • *CertStoreContainerId* • *NewContainerName* and *NewContainerType*
KeystoreIds	Body	**Required**. An array of strings indicating the certificate store GUIDs for the stores you want to place into the container.
NewContainerName	Body	**Required***. A string that sets the name of the container if you would like to create a new container while assigning store(s) to it. One of the following is **required**: • *CertStoreContainerId* • *NewContainerName* and *NewContainerType*
NewContainerType	Body	**Required***. An integer for the container type if you would like to create a new container while assigning store(s) to it. Container types match certificate store types. Use the *GET /CertificateStoreTypes* method with a query (e.g. storetype -eq 7) or *GET /CertificateStoreTypes/{id}* method to determine what a particular certificate store type ID maps to. For example, type 2 maps to *PEM File* and type 10 maps to *F5 SSL Profiles REST*. One of the following is **required**: • *CertStoreContainerId* • *NewContainerName* and *NewContainerType*

Table 333: PUT Certificate Stores Assign Container Response Data

Name	Description
Id	A string indicating the GUID of the certificate store within Keyfactor Command. This ID is automatically set by Keyfactor Command.
ContainerId	An integer indicating the ID of the certificate store's associated certificate store container, if applicable (see GET Certificate Store Containers on page 1477).
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Storepath	A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information. The maximum number of characters supported in this field is 722.
CertStoreInventoryJobId	A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
Approved	A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.
CreateIfMissing	A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.
Properties	Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information). When reading this field, the values are returned as simple key value pairs, with the

Name	Description

	values being individual values. When writing, the values are specified as objects, though they are typically single values.
	For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"`
	However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:
	`"{ \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:
	`"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}}, \"ServerPassword\":{\"value\": {\"SecretValue\":\"MySuperSecretPassword\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`
	Note: There are three standard properties that are used for certificate store

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>types that require server credentials (e.g. F5):
ServerUsername
ServerPassword
ServerUseSsl
These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator for this store.</td></tr>
<tr><td>AgentAssigned</td><td>A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false).</td></tr>
<tr><td>ContainerName</td><td>A string indicating the name of the certificate store’s associated container, if applicable.</td></tr>
<tr><td>InventorySchedule</td><td>An object indicating the inventory schedule for this certificate store. The following schedule types are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>ReenrollmentStatus</td><td>
An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. The following reenrollment fields are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Data</td><td>A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).</td></tr>
<tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.</td></tr>
<tr><td>Message</td><td>A string indicating the reason the certificate store cannot re-enroll, if applicable.</td></tr>
<tr><td>JobProperties</td><td>An array of strings containing the unique entry parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific entry parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is virtualServerName and is returned by GET CertificateStoreTypes like so:
<pre><code>"JobProperties": ["sniCert",
"virtualServerName"]</code></pre>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>It can be seen in the Keyfactor Command Management Portal when editing the certificate store type on the *Entry Parameters* tab. The setting is referenced using the following format: <pre>"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]</pre> This field is optional.</td></tr><tr><td>CustomAliasAllowed</td><td>An integer indicating the option for a custom alias for this certificate store. • 0—forbidden • 1—optional • 2—required</td></tr><tr><td>EntryParameters</td><td>An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include: <table><tr><th>Name</th><th>Description</th></tr><tr><td>StoreTypeID</td><td>**Required**. An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above. If you are updating a certificate store type, this field is **required**.</td></tr><tr><td>Name</td><td>**Required**. A string containing the short name of the entry parameter. If you choose to define an entry parameter, this field is **required**. The name should be entered without spaces.</td></tr><tr><td>DisplayName</td><td>**Required**. A string containing the full display name of the entry para-</td></tr></table></td></tr></table>

Name	Description
	Name / **Description** Name / Description meter. If you choose to define an entry parameter, this field is **required**. Type: **Required**. A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret If you choose to define an entry parameter, this field is **required**. RequiredWhen: An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. The default is *false*. • **OnRemove**: If set to *true*, a value must be provided for this

Name	Description
	Name / **Description**
	Name / Description
	field when configuring a remove certificate job. The default is *false*. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*.
	DependsOn / A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.
	DefaultValue / A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*. This value is unset by default.
	Options / A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if *Type* is set to *MultipleChoice*. This value is unset by default.
	For example, to set a multiple choice entry parameter:

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><pre>"EntryParameter": [
	   {
	      "StoreTypeId": 111,
	      "Name": "ZooAnimal",
	      "DisplayName": "Favorite Zoo Animal",
	      "Type": "MultipleChoice",
	      "RequiredWhen": {
	         "HasPrivateKey": false,
	         "OnAdd": true,
	         "OnRemove": true,
	         "OnReenrollment": true
	      },
	      "DefaultValue": "Penguin",
	      "Options": "Tiger-
	,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
	   }
	]</pre>
	This value is unset by default.
	
	  **Tip:** What's the difference between properties (custom fields) and entry parameters?
	• Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record.
	• Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that</td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).</td></tr>
</table>
</td></tr>
<tr><td>SetNewPasswordAllowed</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false).</td></tr>
<tr><td>Password</td><td>Note: Secret data is stored in the secrets table or a PAM provider and is not returned in responses.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.14 POST Certificate Stores Approve

The POST /CertificateStores/Approve method is used to approve one or more certificate stores currently in the pending state—having been discovered using the certificate store discover option (see PUT Certificate Stores Discovery Job on page 1434). If more than one certificate store is included in the array, all stores must be of the same store type (e.g. Java keystore). This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 334: POST Certificate Stores Approve Input Parameters

Name	In	Description
Id	Body	**Required**. The GUID of the pending certificate store. Use the *GET /CertificateStores* method (see GET Certificate Stores on page 1327) with a query of "Approved -eq false" to retrieve a list of all your unapproved certificate stores to determine the GUID of the store.
ContainerId	Body	An integer that identifies the container in which the certificate store should be placed on approval. Use the *GET /CertificateStores/Containers* method (see GET Certificate Store Containers on page 1477) to retrieve a list of your defined certificate store containers to determine the container ID to use.
CertStoreType	Body	**Required**. An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.

Properties	Body	**Required***. Some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information).
		When reading this field, the values are returned as simple key value pairs, with the values being individual values. When writing, the values are specified as objects, though they are typically single values.
		For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
		`"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"`
		However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:
		`"{ \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"`
		An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:

Name	In	Description

		<pre>"{
		  \"ServerUsername\":{\"value\":{\"SecretValue\":\"*KEYEXAMPLE\\\\jsmith*\"}},
		  \"ServerPassword\":{\"value\":{\"SecretValue\":\"*MySuperSecretPassword*\"}},
		  \"ServerUseSsl\":{\"value\":\"true\"}
		}"</pre>
		An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):
		<pre>"{
		  \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\":
		{\"SecretId\":\"*MyUserID*\"}}},
		  \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\":
		{\"SecretId\":\"*MyPasswordID*\"}}},
		  \"ServerUseSsl\":{\"value\":\"true\"}
		}"</pre>
		Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):
		• ServerUsername
		• ServerPassword
		• ServerUseSsl
		These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
		This field is **required** for certificate store types that store additional properties in this parameter.

Password	Body	**Required**. An object indicating the source for and details of the credential information Keyfactor Command will use to access the certificates in a specific certificate store (the store password). This is different from credential information Keyfactor Command uses to access a certificate store server as a whole. The former (this setting) is typically used for Java keystores; the latter is typically used for certificates stores in Windows certificate stores and on Citrix NetScaler and F5 devices and set at the server level, not the certificate store level. Certificate stores that require credentials support up to three possible credential options: • Use no store password.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
This option is supported for Java keystores that would normally require a password, but can be configured with the no password option (see Value, below).
• Store the credential information in the Keyfactor secrets table.
A Keyfactor secret is a user-defined password that is encrypted and stored securely in the Keyfactor Command database.
• Load the credential information from a PAM provider.
See Privileged Access Management (PAM) on page 742 and PAM Providers on page 1871 for more information.
The possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string—submitted as an object—indicating a password to be stored as a Keyfactor secret.
Tip: To set the no password option on a store, submit the password with a null value. For example:
<pre>"Password": {
   "SecretValue": {null}
}</pre>
To set the value to a string to be stored in the Keyfactor secrets table, include the password in quotes. For example:
<pre>"Password": {
   "SecretValue":
"MyVerySecurePassword"
}</pre></td></tr>
<tr><td>SecretTypeGuid</td><td>A string indicating the Keyfactor Command reference GUID for the type of credentials. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>InstanceId</td><td>The Keyfactor Command reference ID for the secret provider. If you are using a secret provider with an integer ID, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>InstanceGuid</td><td>The Keyfactor Command reference GUID for the secret provider. If you are using a secret provider with a GUID ID,</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>this will be used. This value is automatically set by Keyfactor Command.</td></tr><tr><td>ProviderTypeParameterValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParams. PAM provider type parameter values include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Value</td><td>A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr><tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used.</td></tr><tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the PAM provider. If you are attaching to something with a GUID ID, this will be used.</td></tr><tr><td>Provider</td><td>An object containing information about the provider. PAM provider details include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider.</td></tr></table></td></tr></table></td></tr></table>

Name	In	Description

Name	Description

Name	Description

Name	Description
Name	A string indicating the internal name for the PAM provider.
Area	An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indicating they are used for certificate stores.
Provider-Type	An array of objects containing details about the provider type for the provider, including:

Nam-e	Descrip-tion
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string that

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Nam-e</th><th>Descrip-tion</th></tr><tr><td></td><td>indicates the name of the provider type.</td></tr><tr><td>Provi-der Type Para-ms</td><td>An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. See below instance of Provider-TypeParam for details.</td></tr></table></td></tr><tr><td>Provider-Type</td><td>An array of objects containing the values for</td></tr></table></td></tr></table></td></tr></table>

Name	In	Description
		Name / **Description** **Name** / **Description** **Name** / **Description** Para-mValues: the provider types specified by Provider-TypeParams. See the previous level of *Provider-TypeParamValues* for details. SecuredAreaId: An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any. This is considered deprecated and may be removed in a future release. Remote: A Boolean indicating whether the *Remote Provider* checkbox is checked when adding a new PAM provider (true), or not (false). See PAM Provider Configuration in Keyfactor Command on page 749. ProviderTypeParam: An array of objects that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. PAM provider type parameters include:

Name	In	Description

Name	Description

Name	Description

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of *false*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of *true*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.
DataType	An integer indicating the data type for the parameter. Possible values are: • 1 = String • 2 = Secret
InstanceLevel	A Boolean that sets whether the parameter is used to

Name	In	Description
		Name — **Description** **Name** — **Description** **Name** — **Description** define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (true). For an example, see GET PAM Providers on page 1898. ProviderType — An object containing details for the provider type. **Name** — **Description** Id — A string indicating the Keyfactor Command reference GUID for the PAM provider type

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>ProviderTypeParams</td><td>Unused field</td></tr></table></td></tr></table></td></tr></table></td></tr><tr><td>Provider</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider.</td></tr><tr><td>IsManaged</td><td>A Boolean indicating whether the credentials for the store are managed by a PAM provider (true) or stored in the Keyfactor secrets table (false). This value is automatically set by Keyfactor Command.</td></tr></table> This field is **required** for Java keystores.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.15 POST Certificate Stores Schedule

The POST /CertificateStores/Schedule method is used to create and assign a schedule to one or more certificate stores in Keyfactor Command. The POST request must contain an array of certificate store GUIDs and the properties that make up the schedule to attach to the store(s). This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/schedule/
OR
/certificate_stores/schedule/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 335: POST Certificate Stores Schedule Input Parameters

Name	In	Description
StoreIds	Body	**Required**. An array of strings providing the certificate store GUIDs to schedule.
Schedule	Body	**Required**. An object indicating the inventory schedule for the certificate store(s). Supported schedules are:

Name	Description
Off	Turn off a previously configured schedule.
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO

Name	In	Description

		Name / **Description**
		Name / **Description**
		8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
		For example, daily at 11:30 pm:
		`"Daily": { "Time": "2023-11-25T23:30:00Z" }`
		ExactlyOnce: A dictionary that indicates a job scheduled to run at the time specified with the parameter:
		Name / Description
		Time: The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
		For example, exactly once at 11:45 am:
		`"ExactlyOnce": { "Time": "2023-11-27T11:45:00Z" }`
		Tip: In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *ExactlyOnce*.
		Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.16 POST Certificate Stores Reenrollment

The POST /CertificateStores/Reenrollment method is used to schedule an existing certificate store for reenrollment. The reenrollment method is available for:

- PEM certificate stores managed by the Native Agent.
- PEM and Java certificate stores managed by Java and Android Agents.
- Any custom certificate store types created to support this functionality.

This endpoint returns 204 with no content upon success. Use the GET /OrchestratorJobs/JobHistory method to check on the progress of the job after submission (see GET Orchestrator Jobs Job History on page 1844).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/enrollment/csr/
/certificate_stores/modify/
OR
/certificates/enrollment/csr/
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

In addition, either the user scheduling the reenrollment job or the user configured to provide authentication to the CA (see Authorization Methods Tab on page 367) must have enrollment permissions configured on the CA and template.

Table 336: POST Certificates Stores Reenrollment Input Parameters

Name	In	Description
KeystoreId	Body	**Required**. The GUID of the certificate store to schedule for reenrollment. Use the *GET /CertificateStores* method (see GET Certificate Stores on page 1327) to retrieve a list of your certificate stores to determine the GUID of the store.
SubjectName	Body	**Required**. A string containing the reenrollment subject name using X.500 format. For example: `"SubjectName": "CN=websrvr14.keyexample.com,OU=IT,O=\"Key Example, Inc.\",L=Independence,ST=OH,C=US"`
AgentGuid	Body	**Required**. The GUID of the orchestrator that is registered with the certificate store. Use the *GET /CertificateStores* method (see GET Certificate Stores on page 1327) to retrieve a list of your certificate stores to determine the GUID of the orchestrator associated with the store.
Alias	Body	**Required**. The alias of the certificate in the certificate store.

JobProperties	Body	An object indicating the unique parameters defined for the certificate store type. The *key* is the name of the specific parameter from the certificate store type definition as returned in the JobProperties on the store type using the *GET CertificateStoreTypes* method and the *value* is the value that should be set for that parameter on a certificate in the certificate store. For example, for NetScaler, the key name that is optionally used to associate a certificate with a virtual server is *NetscalerVserver* and is returned by *GET CertificateStoreTypes* like so:
		`"JobProperties": [ "NetscalerVserver" ]`
		It can be seen in the Keyfactor Command Management Portal when editing the certificate store type in the field for *Management Job Custom Fields*.
		The setting is referenced using the following format:
		`"JobProperties": [ {"NetscalerVserver":"MyVirtualServerName"} ]`
		Note: The only built-in certificate store type that makes use of job properties that can be set on a certificate-by-

Name	In	Description
		certificate basis in the store is NetScaler, which does not support reenrollment. You may have custom certificate store types that make use of this functionality.
CertificateAuthority	Body	A string indicating the certificate authority to which to direct the enrollment request. If this parameter is not provided, the value set in the *Certificate Authority For Submitted CSRs* application setting will be used (see Application Settings: Agents Tab on page 614).
CertificateTemplate	Body	A string indicating the certificate template to use for the enrollment request. If this parameter is not provided, the value set in the *Template For Submitted CSRs* application setting will be used (see Application Settings: Agents Tab on page 614).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.17 POST Certificate Stores Certificates Add

The POST /CertificateStores/Certificates/Add method is used to add a certificate to one or more certificate stores. This method returns HTTP 200 OK on a success with an array of GUIDs for the add jobs. Use the GET /OrchestratorJobs/JobHistory method to check on the progress of the job after submission (see GET Orchestrator Jobs Job History on page 1844).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
/certificate_stores/schedule/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
/certificate_stores/schedule/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. You can use a mixture with, for example, global certificate permissions and container-level certificate store permissions. See Certificate Collection Permissions on page 627 and Container Permissions on page 629 for more information about global vs collection and container permissions.

Table 337: POST Certificate Stores Certificates Add Input Parameters

Name	In	Description
CertificateId	Body	**Required**. An integer containing the Keyfactor Command reference ID of the certificate to be added to the certificate store(s).
CertificateStores	Body	**Required**. An array of objects indicating the certificate store GUIDs to identify the certificate stores to which the certificate should be added and provide appropriate reference information for the certificate in the store. Parameters include: **Name** / **Description** CertificateStoreIds — **Required**. A string containing the GUID for the certificate store to which the certificate should be added. Alias — **Required***. A string providing an alias to be used for the certificate upon entry into the certificate store. The function of the alias varies depending on the certificate store type. For example, for an F5 device, it serves as the file name used to store the file in the device file system, minus the extension (e.g. use alias MyFile for a file named MyFile.pfx) while for a Java keystore, it is stored in the keystore associated with the certificate. Some certificate store types don't require an alias and some do. See Add Certificate on page 74 for more information. This field may be **required** depending on the store type selected. JobFields — An object that sets extra values for the job fields that will be associated with the add job. This option is typically used with custom Any Agent implementations. Overwrite — A Boolean that sets whether a certificate in the store with the *Alias* provided should be overwritten with the certificate being added (true) or not (false). The default is *false*. Use the *GET /Certificates/Locations/{id}* method (see GET Certificates Locations ID on page 1116) to retrieve a list of the locations an existing certificate is in to determine the alias

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>used for the certificate in the certificate store.</td></tr>
<tr><td>EntryPassword</td><td>An object containing the password to set on the entry within the certificate store, if applicable. Only select certificate stores support entry passwords (e.g. Java keystores). Entry password values include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the password. This value only needs to be supplied if you're storing your password in the Keyfactor Command database.</td></tr>
<tr><td>Provider</td><td>An integer that identifies the PAM provider used to store the password. Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use.</td></tr>
<tr><td>Parameters</td><td>An object containing the parameters required by your PAM provider, including the information that identifies the location of the password in the PAM solution. Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of the</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>parameters used by your PAM provider. Only parameters where InstanceLevel is equal to true need to be supplied in the request.
For example, for Delinea (formerly Thycotic), this might be:
<pre>"EntryPassword":
{
  "Provider": 2,
  "Parameters":
{
    "SecretId":
5
  }
},</pre>For CyberArk, this might be:
<pre>"EntryPassword":
{
  "Provider": 5,
  "Parameters":
{
    "Folder":
"Root",
    "Object":
"F5Password"
  }
},</pre></td></tr>
</table>
</td></tr>
<tr><td>PfxPassword</td><td>A string that sets the password to use when saving a certificate with its private key in the certificate store. This is only relevant if there's a</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>private key being added along with the certificate.</td></tr>
<tr><td>IncludePrivateKey</td><td>A Boolean that sets whether to include the private key of the certificate in the certificate store if private keys are optional for the given certificate store (true) or not (false). The default is false.</td></tr>
</table>
For example, to add to one IIS personal store and one NetScaler store without overwriting an existing certificate:
<pre>"CertificateStores": [
   {
      "Alias": "MyCertificate.pfx",
      "CertificateStoreId": "fde12aa7-6643-43db-88e8-5c91c5ce78b3",
      "IncludePrivateKey": true
   },
   {
      "Alias": "C2107973A928859C21330E566B299CD4A0705AE8",
      "CertificateStoreId": "322e12ea-43b2-4aab-80ae-c4ad4569b4e7",
      "IncludePrivateKey": true
   }
]</pre>
</td></tr>
<tr><td>Schedule</td><td>Body</td><td>Required. An object indicating the inventory schedule for the add job. Possible schedule values include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>CollectionId</td><td>Body</td><td>An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.10.18 POST Certificate Stores Certificates Remove

The POST /CertificateStores/Certificates/Remove method is used to remove a certificate from one or more certificate stores. The POST request must contain an array of certificate store GUIDs and the certificate properties that identify the certificate to remove. This method returns HTTP 200 OK on a success with an array of GUIDs for the removal jobs. Use the GET /OrchestratorJobs/JobHistory method to check on the progress of the job after submission (see GET Orchestrator Jobs Job History on page 1844).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
/certificate_stores/schedule/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
/certificate_stores/schedule/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. You can use a mixture with, for example, global certificate permissions and container-level certificate store permissions. See Certificate Collection Permissions on page 627 and Container Permissions on page 629 for more information about global vs collection and container permissions.

Table 338: POST Certificate Stores Certificates Remove Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>CertificateStores</td><td>Body</td><td>Required. An array of objects indicating the certificate store GUIDs and related information to identify the certificate to remove from the certificate store(s). Certificate store detail includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Alias</td><td>Required. A string containing the unique identifier for the certificate in the certificate store. Each type of certificate store has a different format for this alias. Use the GET /Certificates/{id} method (see <a>GET Certificates ID on page 1124) to retrieve the certificate store IDs in which the certificate is stored (CertStoreId) and the aliases under which the certificate is stored in these stores. This information is also available in the certificate details in the Management Portal.</td></tr>
<tr><td>CertificateStoreIds</td><td>Required. A string containing the GUID for the certificate store from which the certificate should be removed.</td></tr>
<tr><td>JobFields</td><td>An object that sets extra values for the job fields that will be associated with the removal job. This option is typically used with custom Any Agent implementations.</td></tr>
</table>
For example, to remove from one IIS personal store and one NetScaler store:
<pre>
"CertificateStores": [
   {
      "Alias": "MyCertificate.pfx",
      "CertificateStoreId": "fde12aa7-6643-43db-88e8-
5c91c5ce78b3"
   },
   {
      "Alias": "C2107973A928859C21330E566B299CD4A0705AE8",
      "CertificateStoreId": "322e12ea-43b2-4aab-80ae-
</pre></td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>

```
c4ad4569b4e7"
   }
]
```

</td></tr>
<tr><td>Schedule</td><td>Body</td><td>

Required. An object indicating the inventory schedule for the removal job. Supported schedules are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).

  Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, exactly once at 11:45 am:


```
"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}
```


  Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>

</td></tr>
</table>

Name	In	Description
		Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
CollectionId	Body	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.11 Certificate Store Containers

The CertificateStoreContainers component of the Keyfactor API provides a set of methods to support management of certificate store containers.

Table 339: Certificate Store Containers Endpoints

Endpoint	Method	Description	Link
/	GET	Returns a list of certificate store containers.	GET Certificate Store Containers on the next page
/	POST	Adds a certificate store container.	POST Certificate Store Containers on page 1479
/{id}	DELETE	Deletes a certificate store container.	DELETE Certificate Store Containers ID on page 1509
/{id}	GET	Returns details for the specified certificate store container.	GET Certificate Store Containers ID on page 1509

Endpoint	Method	Description	Link
/{id}	PUT	Edits a certificate store container.	PUT Certificate Store Containers on page 1484

3.6.11.1 GET Certificate Store Containers

The GET /CertificateStoreContainers method is used to retrieve all certificate store containers. This method returns HTTP 200 OK on a success with container details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/
OR
/certificate_stores/read/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 340: GET Certificate Store Containers Input Parameters

Name	In	Description
PerformRoleCheck	Query	This parameter is not used.
RoleIdList	Query	This parameter is not used.
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, refer to: Using the Containers Search Feature on page 432. The query fields supported for this endpoint are: • CertStoreType (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol) • HasSchedule (True, False) • Id • Name (Short Name)
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 341: GET Certificate Stores Containers Response Data

Name	Description
Id	An integer indicating the ID of the container.
Name	A string indicating the name of the container.
OverwriteSchedules	A Boolean indicating whether the schedule set on the container will overwrite schedules set individually on the certificate stores (true) or not (false).
Schedule	A string containing the inventory schedule set for the container. Schedules are shown in cron syntax. For an interval schedule, this will look like I_mm where mm is the number of minutes (e.g. I_30 for every 30 minutes). For daily schedules, this will look like D_hh:mm where hh:mm is the time to run the job (e.g. D_14:30 for daily at 2:30 pm).
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
StoreCount	An integer indicating the number of stores of the type referenced by CertStoreType in the container.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.11.2 POST Certificate Store Containers

The POST /CertificateStoreContainers method is used to add a new certificate store container. This method returns HTTP 200 OK on a success with container details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 342: POST Certificate Stores Containers Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the container.</td></tr>
<tr><td>Schedule</td><td>Body</td><td>An object containing the inventory schedule set for the container. Schedules are shown in cron syntax. Supported schedules are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
CertStoreType	Body	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+. The default is 0 for a JKS keystore.

Table 343: POST Certificate Stores Containers Response Data

Name	Description
Id	An integer indicating the ID of the container.
Name	A string indicating the name of the container.
Schedule	An object containing the inventory schedule set for the container. Schedules are shown in cron syntax. Supported schedules are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre> "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>CertStoreTyp-e</td><td>An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakey-store, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5We-bServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.11.3 PUT Certificate Store Containers

The PUT /CertificateStoreContainers method is used to edit the specified certificate store container. This method returns HTTP 200 OK on a success with container details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 344: PUT Certificate Store Containers Input Parameters

Name	In	Description
Id	Path	**Required**. An integer indicating the ID of the container.
Name	Body	**Required**. A string indicating the name of the container.
Schedule	Body	An object containing the inventory schedule set for the container. Schedules are shown in cron syntax. Supported schedules are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

Name	In	Description

		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><pre>"Daily": {
		  "Time": "2023-11-25T23:30:00Z"
		}</pre></td></tr></table>
		Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

CertStoreType	Body	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+. The default is 0 for a JKS keystore.

Table 345: PUT Certificate Store Containers Response Data

Name	Description
Id	An integer indicating the ID of the container.
Name	A string indicating the name of the container.
Schedule	An object containing the inventory schedule set for the container. Schedules are shown in cron syntax. Supported schedules are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre> "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>CertStoreType</td><td>An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.</td></tr>
<tr><td>CertificateStores</td><td>An array of objects indicating the certificate store data for the certificate stores within this container. Certificate store details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the GUID of the certificate store within Keyfactor Command.</td></tr>
<tr><td>DisplayName</td><td>A string indicating the display name of the certificate store.</td></tr>
<tr><td>ContainerId</td><td>An integer indicating the ID of the certificate store’s associated certificate store container.</td></tr>
<tr><td>ClientMachine</td><td>A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See <a>Adding or Modifying a Certificate Store on page 413 for more information.</td></tr>
<tr><td>Storepath</td><td>A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See <a>Adding or Modifying a Certificate Store on</td></tr>
</table>
</td></tr>
</table>

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>page 413 in the *Keyfactor Command Reference Guide* for more information.</td></tr><tr><td>CertStoreInventoryJobId</td><td>A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.</td></tr><tr><td>CertStoreType</td><td>An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.</td></tr><tr><td>Approved</td><td>A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.</td></tr><tr><td>CreateIfMissing</td><td>A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.</td></tr><tr><td>Properties</td><td>A string containing additional properties for the container. Only some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information).
	When reading this field, the values are returned as simple key value pairs, with the values being individual values. When writing, the values are specified as objects, though they are typically single values.
	For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be:
	<pre>"{
	  \"privateKeyPath\":\"/opt/app/mystore.key\",
	  \"separatePrivateKey\":\"true\"
	}"</pre></td></tr></table>

Name	Description

Name	Description

However, the syntax used when updating the properties sets the value as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:

```
"{
   \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"},
   \"separatePrivateKey\":{\"value\":\"true\"}
}"
```

An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:

```
"{
   \"ServerUsername\":{\"value\":
{\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}},
   \"ServerPassword\":{\"value\":{\"SecretValue\":\"MySu-
perSecretPassword\"}},
   \"ServerUseSsl\":{\"value\":\"true\"}
}"
```

An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):

```
"{
   \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Para-
meters\":{\"SecretId\":\"MyUserID\"}}},
   \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Para-
meters\":{\"SecretId\":\"MyPasswordID\"}}},
   \"ServerUseSsl\":{\"value\":\"true\"}
}"
```

Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):

- ServerUsername
- ServerPassword
- ServerUseSsl

Name	Description

Name	Description
	These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
AgentId	A string indicating the Keyfactor Command GUID of the orchestrator for this store.
AgentAssigned	A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false).
ContainerName	A string indicating the name of the certificate store's associated container.
InventorySchedule	An object containing the inventory schedule for this certificate store.
ReenrollmentStatus	An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. Reenrollment status information includes:

Name	Description
Data	A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).
AgentId	A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.
Message	A string indicating the reason the certificate store cannot re-enroll, if applicable.
JobProperties	An object containing the unique entry parameters defined for the certificate store type. The *key* is the

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**

	name of the specific parameter from the certificate store type definition as returned in the JobProperties on the store type using the *GET CertificateStoreTypes* method and the *value* is the value that should be set for that parameter on a certificate in the certificate store.
	For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is *virtualServerName* and is returned by *GET CertificateStoreTypes* like so:
	`"JobProperties": [ "sniCert", "virtualServerName"]`
	It can be seen in the Keyfactor CommandManagement Portal when editing the certificate store type on the *Entry Parameters* tab.
	The setting is referenced using the following format:
	`"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]`
	This field is optional.

	CustomAliasAllowed: An integer indicating the option for a custom alias for this certificate store. • 0—forbidden • 1—optional • 2—required
	EntryParameters: An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:

Name	Description

Name	Description

Name	Description

Name	Description
StoreTypeID	An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above.
Name	A string containing the short name of the entry parameter.
DisplayName	A string containing the full display name of the entry parameter.
Type	A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret
RequiredWhen	An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to *true*, a value must

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Name / Description
	be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job.

Name	Description

Name	Description

Name	Description

Name	Description
DependsOn	A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.
DefaultValue	A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*.
Options	A string containing a comma-separated list of multiple choice options for this entry parameter.

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>**Tip:** What's the difference between properties (custom fields) and entry parameters? • Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record. • Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).</td></tr></table></td></tr><tr><td>SetNewPasswordAllowed</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false).</td></tr></table>

Name	Description

Name	Description
Password	An object indicating the source for and details of the credential information Keyfactor Command will use to access the certificates in a specific certificate store (the store password). This is different from credential information Keyfactor Command uses to access a certificate store server as a whole. The former (this setting) is typically used for Java keystores; the latter is typically used for certificates stores on Citrix NetScaler and F5 devices and set at the server level, not the certificate store level (see POST Certificate Stores Server on page 1421). Certificate stores that require credentials support up to three possible credential options: • Use no store password. This option is supported for Java keystores that would normally require a password, but can be configured with the no password option (see *Value*, below). • Store the credential information in the Keyfactor secrets table. A Keyfactor secret is a user-defined password that is encrypted and stored securely in the Keyfactor Command database. • Load the credential information from a PAM provider. See Privileged Access Management (PAM) on page 742 and PAM Providers on page 1871 for more information. The possible values are:

Name	Description
SecretValue	A string—submitted as an object—indicating a password to be stored as a Keyfactor secret. **Tip:** To set the no password option on a store, submit the password with a null value. For example: `"Password": { "SecretValue": {null} }` To set the value to a string to be stored in the Keyfactor secrets table, include the password in quotes. For example:

Name	Description

Name	Description

Name	Description
	 `"Password": { "SecretValue": "MyVerySecurePassword" }`
SecretTypeGuid	A string indicating the Keyfactor Command reference GUID for the type of credentials. This value is automatically set by Keyfactor Command.
InstanceId	An integer indicating the Keyfactor Command reference ID for the secret provider. If you are using a secret provider with an integer ID, this will be used. This value is automatically set by Keyfactor Command.
InstanceGuid	A string indicating the Keyfactor Command reference GUID for the secret provider. If you are using a secret provider with a GUID ID, this will be used. This value is automatically set by Keyfactor Command.
ProviderType ParameterValues	An array of objects containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Value	A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).

Name	Description

Name	Description

Name	Description

Name	Description
InstanceId	An integer indicating the Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used.
InstanceGuid	A string indicating the Keyfactor Command reference GUID for the PAM provider. If you are attaching to something with a GUID ID, this will be used.
Provider	An object containing information about the provider. PAM provider details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider.
Name	A string indicating the internal name for the PAM provider.
Area	An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indic-

Name	Description

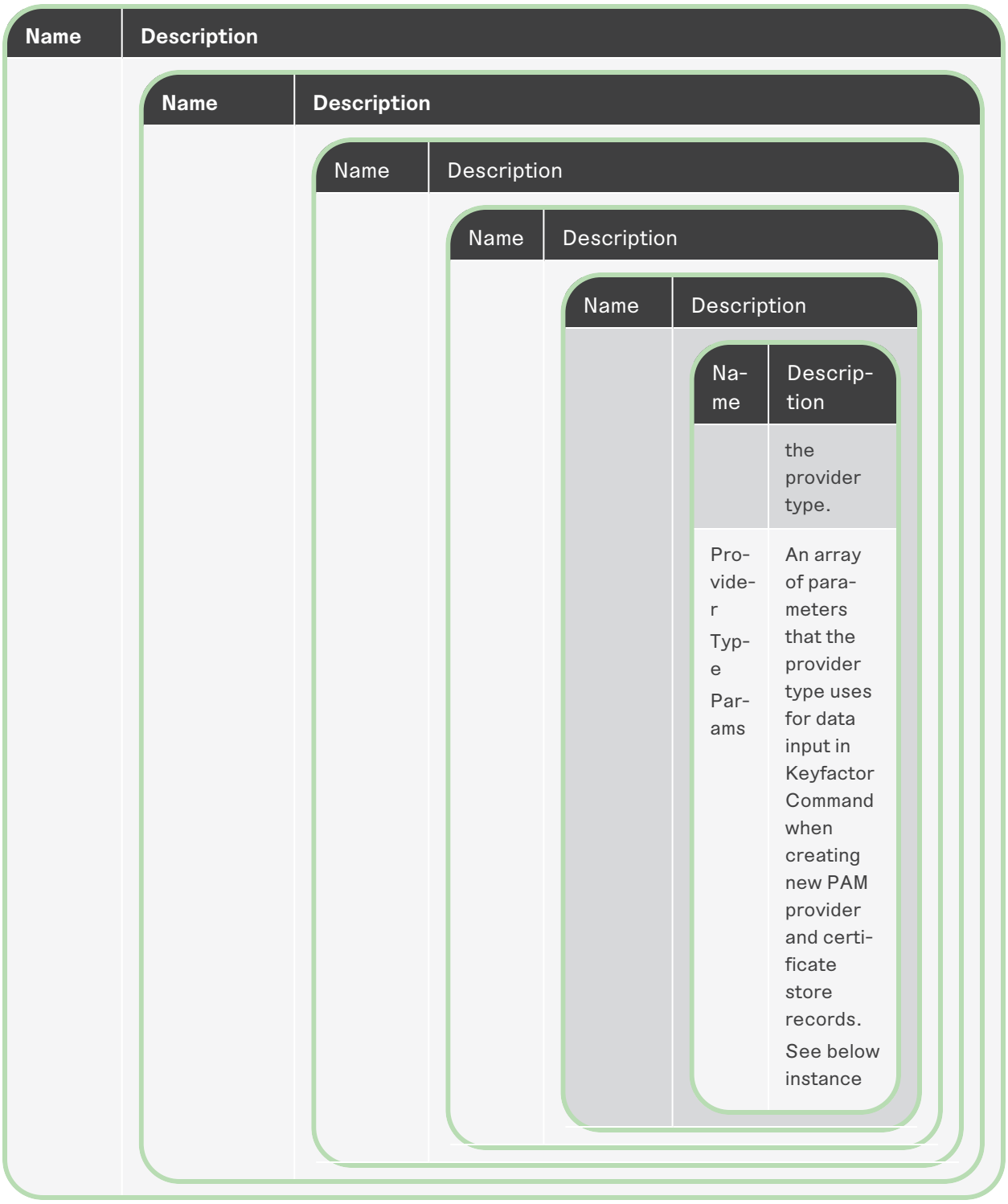
Name	Description

Name	Description

Name	Description

Name	Description
	ating they are used for certificate stores.
ProviderType	An array of objects containing details about the provider type for the provider, including:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string that indicates the name of


Name
Description
Name
Description
Name
Description
Name
Description
Name
Description
Name
Description
the provider type.
ProviderTypeParams
An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records.
See below instance

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
ProviderTypeParamValues	An array of objects containing the values for the provider types specified by ProviderTypeParams. See the previous level of *ProviderTypeParamValues* for details.
SecuredAreaId	An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any. This is considered deprecated and may be

Name	Description
	of *ProviderTypeParam* for details.

Name	Description
	Name / **Description**
	Name / Description
	Name / Description
	Name / Description
	removed in a future release.
	Remote — A Boolean indicating whether the *Remote Provider* checkbox is checked when adding a new PAM provider (true), or not (false). See PAM Provider Configuration in Keyfactor Command on page 749.
	ProviderTypeParam — An array of objects that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. PAM provider type parameters include:
	Name / Description
	Id — An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of *false*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of *true*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.
DataType	An integer indicating the data type for the parameter. Possible values are: • 1 = String • 2 = Secret

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
InstanceLevel	A Boolean that sets whether the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (true). For an example, see GET PAM Providers on page 1898.
ProviderType	An object containing details for the provider type.

Name	Description
Id	A string indicating the

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	Keyfactor Command reference GUID - for the PAM provider type parameter.
Name	A string indicating the intern-

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	al name for the PAM provider type parameter.
ProviderTypeParams	Unused field

ProviderId	An integer indicating the Keyfactor Command reference ID for the PAM provider.
IsManaged	A Boolean indicating whether the credentials for the store are managed by a PAM provider (true) or stored in the Keyfactor secrets table (false). This value is automatically set by Keyfactor Command.

Note: Secret data is stored in the secrets table or a PAM provider and is not returned in responses.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.11.4 DELETE Certificate Store Containers ID

The DELETE /CertificateStoreContainers/{id} method is used to delete the certificate store container with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/
OR
/certificate_stores/modify/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 346: DELETE Certificate Store Containers {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string containing the ID of the certificate store container to delete. Use the *GET /CertificateStoreContainers* method (see GET Certificate Store Containers on page 1477) to retrieve a list of all the certificate store containers to determine the certificate store container ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.11.5 GET Certificate Store Containers ID

The GET /CertificateStoreContainers/{id} method is used to retrieve the certificate store container with the specified ID. This method returns HTTP 200 OK on a success with container details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/
OR
/certificate_stores/read/#/ (where # is a reference to a specific certificate store container ID)
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 347: GET Certificate Store Containers {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string containing the ID of the certificate store container. Use the *GET /CertificateStoreContainers* method (see GET Certificate Store Containers on page 1477) to retrieve a list of all the certificate store containers to determine the certificate store container ID.

Table 348: GET Certificate Stores Containers {id} Response Data

Name	Description
Id	An integer indicating the ID of the container.
Name	A string indicating the name of the container.
Schedule	A string containing the inventory schedule set for the container. Schedules are shown in cron syntax. For an interval schedule, this will look like I_mm where mm is the number of minutes (e.g. I_30 for every 30 minutes). For daily schedules, this will look like D_hh:mm where hh:mm is the time to run the job (e.g. D_14:30 for daily at 2:30 pm).
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
CertificateStores	An array of objects indicating the certificate store data for the certificate stores within this container. Certificate store details include:

Name	Description
Id	A string indicating the GUID of the certificate store within Keyfactor Command.
DisplayName	A string indicating the display name of the certificate store.
ContainerId	An integer indicating the ID of the certificate store's associated certificate store container.
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Storepath	A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 in the *Keyfactor Command Reference Guide* for more information.

Name	Description

Name	Description
CertStoreInventoryJobId	A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. Built-in certificates store types are: (0-Javakeystore, 2-PEMFile, 3-F5SSLProfiles,4-IISRoots, 5-NetScaler, 6-IISPersonal, 7-F5WebServer, 8-IISRevoked, 9-F5WebServerREST, 10-F5SSLProfilesREST, 11-F5CABundlesREST, 100-AmazonWebServices, 101-FileTransferProtocol). Any custom extensions for the Keyfactor Universal Orchestrator you add will have certificate store types numbered 102+.
Approved	A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.
CreateIfMissing	A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.
Properties	A string containing additional properties for the container. Only some types of certificate stores have additional properties that are stored in this parameter. The data is stored in a series of, typically, key value pairs that define the property name and value (see GET Certificate Store Types on page 1546 for more information). When reading this field, the values are returned as simple key value pairs, with the values being individual values. When writing, the values are specified as objects, though they are typically single values. For example, on a GET request for a *PEM* store configured with a separate private key, the contents of this field might be: `"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"` However, the syntax used when updating the properties sets the value

Name	Description

Name	Description

as a key value pair using *value* as the key. For example, on a POST or PUT request for a *PEM* store configured with a separate private key, the contents of this field might be:

```
"{
   \"privateKeyPath\":{\"value\":\"/opt/app/mystore.key\"},
   \"separatePrivateKey\":{\"value\":\"true\"}
}"
```

An example server properties parameter POST for an F5 or Citrix NetScaler store would contain:

```
"{
   \"ServerUsername\":{\"value\":
{\"SecretValue\":\"KEYEXAMPLE\\\\jsmith\"}},
   \"ServerPassword\":{\"value\":{\"SecretValue\":\"MySu-
perSecretPassword\"}},
   \"ServerUseSsl\":{\"value\":\"true\"}
}"
```

An example server properties parameter POST for an F5 or Citrix NetScaler store with the username and password stored as PAM secrets would contain (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898):

```
"{
   \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Para-
meters\":{\"SecretId\":\"MyUserID\"}}},
   \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Para-
meters\":{\"SecretId\":\"MyPasswordID\"}}},
   \"ServerUseSsl\":{\"value\":\"true\"}
}"
```

Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):

- ServerUsername
- ServerPassword
- ServerUseSsl

These replace the separate certificate store server records that

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.</td></tr><tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator for this store.</td></tr><tr><td>AgentAssigned</td><td>A Boolean that indicates whether there is an orchestrator assigned to this certificate store (true) or not (false).</td></tr><tr><td>ContainerName</td><td>A string indicating the name of the certificate store's associated container.</td></tr><tr><td>InventorySchedule</td><td>An object containing the inventory schedule for this certificate store.</td></tr><tr><td>ReenrollmentStatus</td><td>An object that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job. Reenrollment status information includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Data</td><td>A Boolean that indicates whether the certificate store can use the re-enrollment function (true) or not (false).</td></tr><tr><td>AgentId</td><td>A string indicating the Keyfactor Command GUID of the orchestrator that can re-enroll the certificate store.</td></tr><tr><td>Message</td><td>A string indicating the reason the certificate store cannot re-enroll, if applicable.</td></tr><tr><td>JobProperties</td><td>An object containing the unique entry parameters defined for the certificate store type. The key is the name of the specific parameter from the certificate</td></tr></table></td></tr></table>

Name	Description
	Name / **Description**

	store type definition as returned in the JobProperties on the store type using the *GET CertificateStoreTypes* method and the *value* is the value that should be set for that parameter on a certificate in the certificate store.
	For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is *virtualServerName* and is returned by *GET CertificateStoreTypes* like so:
	`"JobProperties": [ "sniCert", "virtualServerName"]`
	It can be seen in the Keyfactor CommandManagement Portal when editing the certificate store type on the *Entry Parameters* tab.
	The setting is referenced using the following format:
	`"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]`
	This field is optional.

	CustomAliasAllowed: An integer indicating the option for a custom alias for this certificate store. • 0—forbidden • 1—optional • 2—required
	EntryParameters: An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Name / Description
	StoreTypeID — An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above.
	Name — A string containing the short name of the entry parameter.
	DisplayName — A string containing the full display name of the entry parameter.
	Type — A string containing the type of the entry parameter: • String • Bool • MultipleChoice • Secret
	RequiredWhen — An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: • **HasPrivateKey**: If set to *true*, a value must

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Name / Description
	be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. • **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job.

Name	Description

Name	Description

Name	Description

Name	Description
DependsOn	A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.
DefaultValue	A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*.
Options	A string containing a comma-separated list of multiple choice options for this entry parameter.

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>**Tip:** What's the difference between properties (custom fields) and entry parameters? • Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record. • Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).</td></tr></table></td></tr><tr><td>SetNewPasswordAllowed</td><td>A Boolean that indicates whether the store password can be changed (true) or not (false).</td></tr></table>

Name	Description

Name	Description
Password	An object indicating the source for and details of the credential information Keyfactor Command will use to access the certificates in a specific certificate store (the store password). This is different from credential information Keyfactor Command uses to access a certificate store server as a whole. The former (this setting) is typically used for Java keystores; the latter is typically used for certificates stores on Citrix NetScaler and F5 devices and set at the server level, not the certificate store level (see POST Certificate Stores Server on page 1421). Certificate stores that require credentials support up to three possible credential options: • Use no store password. This option is supported for Java keystores that would normally require a password, but can be configured with the no password option (see *Value*, below). • Store the credential information in the Keyfactor secrets table. A Keyfactor secret is a user-defined password that is encrypted and stored securely in the Keyfactor Command database. • Load the credential information from a PAM provider. See Privileged Access Management (PAM) on page 742 and PAM Providers on page 1871 for more information. The possible values are:

Name	Description
SecretValue	A string—submitted as an object—indicating a password to be stored as a Keyfactor secret. **Tip:** To set the no password option on a store, submit the password with a null value. For example: `"Password": { "SecretValue": {null} }` To set the value to a string to be stored in the Keyfactor secrets table, include the password in quotes. For example:

Name	Description

Name	Description

Name	Description
	 "Password": { "SecretValue": "MyVerySecurePassword" }
SecretTypeGuid	A string indicating the Keyfactor Command reference GUID for the type of credentials. This value is automatically set by Keyfactor Command.
InstanceId	An integer indicating the Keyfactor Command reference ID for the secret provider. If you are using a secret provider with an integer ID, this will be used. This value is automatically set by Keyfactor Command.
InstanceGuid	A string indicating the Keyfactor Command reference GUID for the secret provider. If you are using a secret provider with a GUID ID, this will be used. This value is automatically set by Keyfactor Command.
ProviderType ParameterValues	An array of objects containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Value	A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).

Name	Description

Name	Description

Name	Description

Name	Description
InstanceId	An integer indicating the Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used.
InstanceGuid	A string indicating the Keyfactor Command reference GUID for the PAM provider. If you are attaching to something with a GUID ID, this will be used.
Provider	An object containing information about the provider. PAM provider details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider.
Name	A string indicating the internal name for the PAM provider.
Area	An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indic-

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	ating they are used for certificate stores.
ProviderType	An array of objects containing details about the provider type for the provider, including:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string that indicates the name of

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	the provider type.
Provider Type Params	An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. See below instance

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
ProviderTypeParamValues	An array of objects containing the values for the provider types specified by ProviderTypeParams. See the previous level of *ProviderTypeParamValues* for details.
SecuredAreaId	An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any. This is considered deprecated and may be

Name	Description
	of *ProviderTypeParam* for details.

Name	Description
	Name / **Description**
	Name / Description
	Name / Description
	Name / Description
	Name / Description
	removed in a future release.
	Remote — A Boolean indicating whether the *Remote Provider* checkbox is checked when adding a new PAM provider (true), or not (false). See PAM Provider Configuration in Keyfactor Command on page 749.
	ProviderTypeParam — An array of objects that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. PAM provider type parameters include:
	Name / Description
	Id — An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of *false*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of *true*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.
DataType	An integer indicating the data type for the parameter. Possible values are: • 1 = String • 2 = Secret

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
InstanceLevel	A Boolean that sets whether the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (true). For an example, see GET PAM Providers on page 1898.
ProviderType	An object containing details for the provider type.

Name	Description
Id	A string indicating the

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	Keyfactor Command reference GUID - for the PAM provider type parameter.
Name	A string indicating the intern-

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	al name for the PAM provider type parameter.
ProviderTypeParams	Unused field

ProviderId	An integer indicating the Keyfactor Command reference ID for the PAM provider.
IsManaged	A Boolean indicating whether the credentials for the store are managed by a PAM provider (true) or stored in the Keyfactor secrets table (false). This value is automatically set by Keyfactor Command.

Note: Secret data is stored in the secrets table or a PAM provider and is not returned in responses.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.12 Certificate Store Types

CertificateStoreTypes define constraints and properties of different kinds of certificates stores. Keyfactor Command contains default certificate store types and also allows users to define certificate store types for custom certificate stores.

Table 349: Certificate Store Type Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes a certificate store type using StoreType number.	DELETE Certificate Store Types ID below
/{id}	GET	Returns certificate store type details for the specified certificate store type using StoreType number.	GET Certificate Store Types ID on the next page
/Name/{name}	GET	Returns certificate store type details for the specified certificate store type using ShortName.	GET CertificateStoreTypes Name Name on page 1538
/	DELETE	Delete multiple certificate store types using StoreType number.	DELETE Certificate Store Types on page 1545
/	GET	Returns all certificate store types with paging and options to the specified detail level.	GET Certificate Store Types on page 1546
/	POST	Creates a new certificate store type.	POST Certificate Store Types on page 1552
/	PUT	Updates a certificate store type using StoreType number.	PUT Certificate Store Types on page 1566

3.6.12.1 DELETE Certificate Store Types ID

The DELETE /CertificateStoreTypes/{id} method is used to delete an existing certificate store type with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/

Table 350: DELETE Certificate Store Types {id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the certificate store type to delete. Use the *GET /CertificateStoreTypes* method (see GET Certificate Store Types on page 1546) to retrieve a list of all the certificate store types to determine the certificate store type ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.12.2 GET Certificate Store Types ID

The GET /CertificateStoreTypes/{id} method is used to return the certificate store type with the specified ID. This method returns HTTP 200 OK on a success with details for the certificate store type specified.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/ OR container permission

Table 351: GET Certificate Store Types {id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the certificate store type. Use the *GET /CertificateStoreTypes* method (see GET Certificate Store Types on page 1546) to retrieve a list of all the certificate store types to determine the certificate store type ID.

Table 352: GET Certificate Store Types {id} Response Data

Name	Description
Name	A string containing the full name of the certificate store type.
ShortName	A string containing the short name assigned to the certificate store type.
Capability	A string containing a reference name for the certificate store type (e.g. NS for a NetScaler store).
StoreType	An integer indicating the Keyfactor Command reference ID for the certificate store type. The ID is automatically assigned by Keyfactor Command.
ImportType	An integer that indicates the import type for the certificate store type. The ID is automatically assigned by Keyfactor Command and generally matches the *StoreType* for custom certificate store types.
LocalStore	A Boolean that indicates whether the store is local to the orchestrator machine (true) as, for example, JKS and PEM stores managed by the Keyfactor Java Agent or remote (false) as, for example, IIS stores managed by the Keyfactor Universal Orchestrator.
SupportedOperations	An object containing Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of objects indicating unique parameters for the certificate store type. In the Keyfactor Command Management Portal these are known as *Custom Fields*. Property parameters include:

Name	Description
Name	A string containing the short name of the property.
DisplayName	A string containing the full display name of the property.
Type	A string containing the type of the property: • String • Bool

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• MultipleChoice
• Secret</td></tr>
<tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value(s) of the parameter. If Type is Multiple Choice, this field should contain a comma-separated list of multiple choice options for this parameter. If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that indicates whether the parameter is required (true) or not (false).</td></tr>
</table>

Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):
• ServerUsername
• ServerPassword
• ServerUseSsl

These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
</td></tr>
<tr><td>PasswordOptions</td><td>An object indicating options for the password in the certificate store type. Password options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EntrySupported</td><td>A Boolean that indicates whether entry of a password for the certificate in the certificate store is</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>allowed (true) or not (false).</td></tr>
<tr><td>StoreRequired</td><td>A Boolean that indicates whether entry of a password on the certificate store as a whole is required (true) or not (false).</td></tr>
<tr><td>Style</td><td>A string containing the style of password:
Default: Keyfactor Command will randomly generate a password.
Custom: Allow a password to be entered and authenticated when enrolling for a certificate through Keyfactor Command when installing to a store of this type. The Custom option can be selected only if the Allow Custom Password in the Application Settings, is equal to True. For more details, see Application Settings: Enrollment Tab on page 609.</td></tr>
</table>
</td></tr>
<tr><td>StorePathValue</td><td>A string containing the value(s) for the certificate store path.</td></tr>
<tr><td>PrivateKeyAllowed</td><td>A string containing the option for private key requirements for certificates stored in stores with this certificate store type:
Forbidden: Private key is not required; generally, applies to trust stores (e.g. Root CA certificates).
Optional: Private key is optional; applies to store types that could represent either a Trust Store or End-Entity Store.
Required: Private key is required; applies to stores that hold an End-Entity Certificate (server or client authorization).</td></tr>
<tr><td>ServerRequired</td><td>A Boolean that indicates whether server access is required for adding certificate stores for this certificate store type (true) or not (false). If set to true, a user will be prompted for a username and password to connect to the remote server.</td></tr>
<tr><td>PowerShell</td><td>A Boolean that indicates whether jobs for the store type are implemented using PowerShell (true) instead of a .NET class or not (false).</td></tr>
<tr><td>BlueprintAllowed</td><td>A Boolean that indicates whether certificate stores of this type will be included when creating or applying blueprints. For more details, see Orchestrator Blueprints on page 521.</td></tr>
<tr><td>CustomAliasAllowed</td><td>A string containing the selected certificate store type alias option:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Forbidden: A custom alias is not required and cannot be supplied.
Optional: A custom alias is optional.
Required: A custom alias is required.

The certificate store alias serves as an identifier for the certificate in the store. Depending on the type of store, it may be a file name, a certificate thumbprint, a string reference, or some other information. Some types of stores may not support associating an alias with the certificate (e.g. IIS trusted root).
</td></tr>
<tr><td>EntryParameters</td><td>
An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr>
<tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr>
<tr><td>Type</td><td>A string containing the type of the entry parameter:

String
Bool
MultipleChoice
Secret

</td></tr>
<tr><td>RequiredWhen</td><td>An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are:

HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store.
OnAdd: If set to true, a value must be provided for this field when configuring an add certi-

</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>ficate job. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job.</td></tr><tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*.</td></tr><tr><td>Options</td><td>A string containing a comma-separated list of multiple choice options for this entry parameter.</td></tr></table> **Tip:** What's the difference between properties (custom fields) and entry parameters? • Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record. • Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers

Name	Description
	with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).
InventoryEndpoint	A string containing the orchestrator endpoint to which inventory updates are sent.
InventoryJobType	A GUID identifying the job type for inventory jobs.
ManagementJobType	A GUID identifying the job type for management jobs.
DiscoveryJobType	A GUID identifying the job type for discovery jobs.
EnrollmentJobType	A GUID identifying the job type for reenrollment jobs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.12.3 GET CertificateStoreTypes Name Name

The GET /CertificateStoreTypes/Name/{name} method is used to return the certificate store type with the specified short name. This method returns HTTP 200 OK on a success with details for the certificate store type specified.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/

Table 353: GET Certificate Store Types Name {Name} Input Parameters

Name	In	Description
name	Path	**Required**. The short name of the certificate store type. Use the *GET /CertificateStoreTypes* method (see GET Certificate Store Types on page 1546) to retrieve a list of all the certificate store types to determine the certificate store type short name.

Table 354: GET Certificate Store Types Name {Name} Response Data

Name	Description
Name	A string containing the full name of the certificate store type.
ShortName	A string containing the short name assigned to the certificate store type.
Capability	A string containing a reference name for the certificate store type (e.g. NS for a NetScaler store).
StoreType	An integer indicating the Keyfactor Command reference ID for the certificate store type. The ID is automatically assigned by Keyfactor Command.
ImportType	An integer that indicates the import type for the certificate store type. The ID is automatically assigned by Keyfactor Command and generally matches the *StoreType* for custom certificate store types.
LocalStore	A Boolean that indicates whether the store is local to the orchestrator machine (true) as, for example, JKS and PEM stores managed by the Keyfactor Java Agent or remote (false) as, for example, IIS stores managed by the Keyfactor Universal Orchestrator.
SupportedOperations	An object containing Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of objects indicating unique parameters for the certificate store type. In the Keyfactor Command Management Portal these are known as *Custom Fields*. Property parameters include:

Name	Description
Name	A string containing the short name of the property.
DisplayName	A string containing the full display name of the property.
Type	A string containing the type of the property: • String • Bool

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• MultipleChoice
• Secret</td></tr>
<tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value(s) of the parameter. If Type is Multiple Choice, this field should contain a comma-separated list of multiple choice options for this parameter. If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that indicates whether the parameter is required (true) or not (false).</td></tr>
</table>

Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):
• ServerUsername
• ServerPassword
• ServerUseSsl

These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
</td></tr>
<tr><td>PasswordOptions</td><td>An object indicating options for the password in the certificate store type. Password options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EntrySupported</td><td>A Boolean that indicates whether entry of a password for the certificate in the certificate store is</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>allowed (true) or not (false).</td></tr>
<tr><td>StoreRequired</td><td>A Boolean that indicates whether entry of a password on the certificate store as a whole is required (true) or not (false).</td></tr>
<tr><td>Style</td><td>A string containing the style of password:
Default: Keyfactor Command will randomly generate a password.
Custom: Allow a password to be entered and authenticated when enrolling for a certificate through Keyfactor Command when installing to a store of this type. The Custom option can be selected only if the Allow Custom Password in the Application Settings, is equal to True. For more details, see Application Settings: Enrollment Tab on page 609.</td></tr>
</table>
</td></tr>
<tr><td>StorePathValue</td><td>A string containing the value(s) for the certificate store path.</td></tr>
<tr><td>PrivateKeyAllowed</td><td>A string containing the option for private key requirements for certificates stored in stores with this certificate store type:
Forbidden: Private key is not required; generally, applies to trust stores (e.g. Root CA certificates).
Optional: Private key is optional; applies to store types that could represent either a Trust Store or End-Entity Store.
Required: Private key is required; applies to stores that hold an End-Entity Certificate (server or client authorization).</td></tr>
<tr><td>ServerRequired</td><td>A Boolean that indicates whether server access is required for adding certificate stores for this certificate store type (true) or not (false). If set to true, a user will be prompted for a username and password to connect to the remote server.</td></tr>
<tr><td>PowerShell</td><td>A Boolean that indicates whether jobs for the store type are implemented using PowerShell (true) instead of a .NET class or not (false).</td></tr>
<tr><td>BlueprintAllowed</td><td>A Boolean that indicates whether certificate stores of this type will be included when creating or applying blueprints. For more details, see Orchestrator Blueprints on page 521.</td></tr>
<tr><td>CustomAliasAllowed</td><td>A string containing the selected certificate store type alias option:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Forbidden: A custom alias is not required and cannot be supplied.
Optional: A custom alias is optional.
Required: A custom alias is required.

The certificate store alias serves as an identifier for the certificate in the store. Depending on the type of store, it may be a file name, a certificate thumbprint, a string reference, or some other information. Some types of stores may not support associating an alias with the certificate (e.g. IIS trusted root).
</td></tr>
<tr><td>EntryParameters</td><td>
An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr>
<tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr>
<tr><td>Type</td><td>A string containing the type of the entry parameter:

String
Bool
MultipleChoice
Secret

</td></tr>
<tr><td>RequiredWhen</td><td>An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are:

HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store.
OnAdd: If set to true, a value must be provided for this field when configuring an add certi-

</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>ficate job. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job.</td></tr><tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*.</td></tr><tr><td>Options</td><td>A string containing a comma-separated list of multiple choice options for this entry parameter.</td></tr></table> **Tip:** What's the difference between properties (custom fields) and entry parameters? • Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record. • Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers

Name	Description
	with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).
InventoryEndpoint	A string containing the orchestrator endpoint to which inventory updates are sent.
InventoryJobType	A GUID identifying the job type for inventory jobs.
ManagementJobType	A GUID identifying the job type for management jobs.
DiscoveryJobType	A GUID identifying the job type for discovery jobs.
EnrollmentJobType	A GUID identifying the job type for reenrollment jobs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.12.4 DELETE Certificate Store Types

The DELETE /CertificateStoreTypes method is used to delete multiple certificate store types in one request. IDs of any certificate store types that could not be deleted are returned in the response body. Delete operations will continue until the entire array of IDs has been processed. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/

Table 355: DELETE Certificate Store Types Input Parameters

Name	In	Description
ids	Body	**Required**. An array of integers indicating the Keyfactor Command certificate store type IDs for certificate store types that should be deleted in the form (without parameter name): `[106,108,109]` Use the *GET /CertificateStoreTypes* method (see GET Certificate Store Types below) to retrieve a list of all the certificate store types to determine the certificate store type IDs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.12.5 GET Certificate Store Types

The GET /CertificateStoreTypes method is used to retrieve a list of all certificate store types. This method returns HTTP 200 OK on a success with details of the certificate store types.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/read/ OR container permission

Table 356: GET Certificate Store Types Input Parameters

Name	In	Description
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.

Table 357: GET Certificate Store Types Response Data

Name	Description
Name	A string containing the full name of the certificate store type.
ShortName	A string containing the short name assigned to the certificate store type.
Capability	A string containing a reference name for the certificate store type (e.g. NS for a NetScaler store).
StoreType	An integer indicating the Keyfactor Command reference ID for the certificate store type. The ID is automatically assigned by Keyfactor Command.
ImportType	An integer that indicates the import type for the certificate store type. The ID is automatically assigned by Keyfactor Command and generally matches the *StoreType* for custom certificate store types.
LocalStore	A Boolean that indicates whether the store is local to the orchestrator machine (true) as, for example, JKS and PEM stores managed by the Keyfactor Java Agent or remote (false) as, for example, IIS stores managed by the Keyfactor Universal Orchestrator.
SupportedOperations	An object containing Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of objects indicating unique parameters for the certificate store type. In the Keyfactor Command Management Portal these are known as *Custom Fields*. Property parameters include:

Name	Description
Name	A string containing the short name of the property.
DisplayName	A string containing the full display name of the property.
Type	A string containing the type of the property: • String • Bool

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• MultipleChoice
• Secret</td></tr>
<tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value(s) of the parameter. If Type is Multiple Choice, this field should contain a comma-separated list of multiple choice options for this parameter. If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that indicates whether the parameter is required (true) or not (false).</td></tr>
</table>

Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):
• ServerUsername
• ServerPassword
• ServerUseSsl

These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
</td></tr>
<tr><td>PasswordOptions</td><td>An object indicating options for the password in the certificate store type. Password options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EntrySupported</td><td>A Boolean that indicates whether entry of a password for the certificate in the certificate store is</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>allowed (true) or not (false).</td></tr>
<tr><td>StoreRequired</td><td>A Boolean that indicates whether entry of a password on the certificate store as a whole is required (true) or not (false).</td></tr>
<tr><td>Style</td><td>A string containing the style of password:
Default: Keyfactor Command will randomly generate a password.
Custom: Allow a password to be entered and authenticated when enrolling for a certificate through Keyfactor Command when installing to a store of this type. The Custom option can be selected only if the Allow Custom Password in the Application Settings, is equal to True. For more details, see Application Settings: Enrollment Tab on page 609.</td></tr>
</table>
</td></tr>
<tr><td>StorePathValue</td><td>A string containing the value(s) for the certificate store path.</td></tr>
<tr><td>PrivateKeyAllowed</td><td>A string containing the option for private key requirements for certificates stored in stores with this certificate store type:
Forbidden: Private key is not required; generally, applies to trust stores (e.g. Root CA certificates).
Optional: Private key is optional; applies to store types that could represent either a Trust Store or End-Entity Store.
Required: Private key is required; applies to stores that hold an End-Entity Certificate (server or client authorization).</td></tr>
<tr><td>ServerRequired</td><td>A Boolean that indicates whether server access is required for adding certificate stores for this certificate store type (true) or not (false). If set to true, a user will be prompted for a username and password to connect to the remote server.</td></tr>
<tr><td>PowerShell</td><td>A Boolean that indicates whether jobs for the store type are implemented using PowerShell (true) instead of a .NET class or not (false).</td></tr>
<tr><td>BlueprintAllowed</td><td>A Boolean that indicates whether certificate stores of this type will be included when creating or applying blueprints. For more details, see Orchestrator Blueprints on page 521.</td></tr>
<tr><td>CustomAliasAllowed</td><td>A string containing the selected certificate store type alias option:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Forbidden: A custom alias is not required and cannot be supplied.
Optional: A custom alias is optional.
Required: A custom alias is required.

The certificate store alias serves as an identifier for the certificate in the store. Depending on the type of store, it may be a file name, a certificate thumbprint, a string reference, or some other information. Some types of stores may not support associating an alias with the certificate (e.g. IIS trusted root).
</td></tr>
<tr><td>EntryParameters</td><td>
An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr>
<tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr>
<tr><td>Type</td><td>A string containing the type of the entry parameter:

String
Bool
MultipleChoice
Secret

</td></tr>
<tr><td>RequiredWhen</td><td>An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are:

HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store.
OnAdd: If set to true, a value must be provided for this field when configuring an add certi-

</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>ficate job. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job.</td></tr><tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*.</td></tr><tr><td>Options</td><td>A string containing a comma-separated list of multiple choice options for this entry parameter.</td></tr></table> **Tip:** What's the difference between properties (custom fields) and entry parameters? • Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record. • Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers

Name	Description
	with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).
InventoryEndpoint	A string containing the orchestrator endpoint to which inventory updates are sent.
InventoryJobType	A GUID identifying the job type for inventory jobs.
ManagementJobType	A GUID identifying the job type for management jobs.
DiscoveryJobType	A GUID identifying the job type for discovery jobs.
EnrollmentJobType	A GUID identifying the job type for reenrollment jobs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.12.6 POST Certificate Store Types

The POST /CertificateStoresTypes method is used to create certificate store types in Keyfactor Command. This method returns HTTP 200 OK on a success with a message body containing a list of certificate store type details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/

Table 358: POST Certificate Store Types Input Parameters

Name	In	Description
Name	Body	**Required**. A string containing the full name of the certificate store type. A unique value must be supplied.
ShortName	Body	**Required**. A string containing the short name assigned to the certificate store type. A unique value must be supplied with a maximum of 10 characters.
Capability	Body	A string containing a reference name for the certificate store type (e.g. NS for a NetScaler store).
LocalStore	Body	A Boolean that indicates whether the store is local to the orchestrator machine (true) as, for example, JKS and PEM stores managed by the Keyfactor Java Agent or remote (false) as, for example, IIS stores managed by the Keyfactor Universal Orchestrator. The default is *false*.
SupportedOperations	Body	An object containing Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove The default for each value is *false*.
Properties	Body	An array of objects indicating unique parameters for the certificate store type. In the Keyfactor Command Management Portal these are known as *Custom Fields*. Property parameters include:

Name	Description
Name	**Required**. A string containing the short name of the property. If you choose to define a property, this field is **required**.
DisplayName	**Required**. A string containing the full display name of the property. If you choose to define a property, this field is **required**.
Type	**Required**. A string containing the type

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>of the property:
String
Bool
MultipleChoice
Secret
If you choose to define a property, this field is required.

Note: This field cannot be modified on an edit.</td></tr>
<tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value(s) of the parameter. If Type is Multiple Choice, this field should contain a comma-separated list of multiple choice options for this parameter. If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that indicates whether the parameter is required (true) or not (false).</td></tr>
</table>
Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):
ServerUsername
ServerPassword
ServerUseSsl
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
For example, to set a multiple choice property:
<pre>"Properties": [
   {
      "Name": "Pets",
      "DisplayName": "Popular Pets",
      "Type": "MultipleChoice",
      "DependsOn": "",
      "DefaultValue": "Cat,Dog,Fish,Rat,Mouse",
      "Required": false
   }
]</pre>
This value is unset by default.
</td></tr>
<tr><td>PasswordOptions</td><td>Body</td><td>
An object indicating options for the password in the certificate store type. Password options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EntrySupported</td><td>A Boolean that indicates whether entry of a password for the certificate in the certificate store is allowed (true) or not (false). The default is false.</td></tr>
<tr><td>StoreRequired</td><td>A Boolean that indicates whether entry of a password on the certificate store as a whole is required (true) or not (false). The default is false.</td></tr>
<tr><td>Style</td><td>A string containing the style of password:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Default: Keyfactor Command will randomly generate a password.
Custom: Allow a password to be entered and authenticated when enrolling for a certificate through Keyfactor Command when installing to a store of this type. The Custom option can be selected only if the Allow Custom Password in the Application Settings, is equal to True. For more details, see <a>Application Settings: Enrollment Tab on page 609.

The default value is Default.
</td></tr>
</table>
</td></tr>
<tr><td>StorePathType</td><td>Body</td><td>A string containing the selected store type:

Freeform: Users are required to enter a path defining the certificate store location.
Fixed: A store path does not apply, generally one store per device (e.g. IIS).
MultipleChoice: Allow a comma separated list of options to be entered that users will be able to select from when defining the certificate store location.

This value is unset by default.</td></tr>
<tr><td>StorePathValue</td><td>Body</td><td>A string containing the value(s) for the certificate store path if the StorePathType is set to Fixed or Multiple Choice.
Multiple choice values should be provided in a bracketed comma-delimited list like so:
<pre><code>"StorePathValue": "
[\"Apple\",\"Cherry\",\"Peach\",\"Pear\"]"</code></pre>
This value is unset by default.</td></tr>
<tr><td>PrivateKeyAllowed</td><td>Body</td><td>A string containing the option for private key requirements for certificates stored in stores with this certificate store type:

Forbidden: Private key is not required; generally, applies to trust stores (e.g. Root CA certificates).
</td></tr>
</table>

Name	In	Description
		• *Optional*: Private key is optional; applies to store types that could represent either a Trust Store or End-Entity Store. • *Required*: Private key is required; applies to stores that hold an End-Entity Certificate (server or client authorization). The default value is *Forbidden*.
ServerRequired	Body	A Boolean that indicates whether server access is required for adding certificate stores for this certificate store type (true) or not (false). If set to true, a user will be prompted for a username and password to connect to the remote server. The default is *false*.
PowerShell	Body	A Boolean that indicates whether jobs for the store type are implemented using PowerShell (true) instead of a .NET class or not (false). The default is *false*.
BlueprintAllowed	Body	A Boolean that indicates whether certificate stores of this type will be included when creating or applying blueprints. For more details, see Orchestrator Blueprints on page 521. The default is *false*.
CustomAliasAllowed	Body	A string containing the selected certificate store type alias option: • *Forbidden*: A custom alias is not required and cannot be supplied. • *Optional*: A custom alias is optional. • *Required*: A custom alias is required. The certificate store alias serves as an identifier for the certificate in the store. Depending on the type of store, it may be a file name, a certificate thumbprint, a string reference, or some other information. Some types of stores may not support associating an alias with the certificate (e.g. IIS trusted root). The default value is *Forbidden*.
EntryParameters	Body	An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include: Name: **Required**. A string containing the

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>short name of the entry parameter. If you choose to define an entry parameter, this field is **required**. The name should be entered without spaces.</td></tr><tr><td>DisplayName</td><td>**Required**. A string containing the full display name of the entry parameter. If you choose to define an entry parameter, this field is **required**.</td></tr><tr><td>Type</td><td>**Required**. A string containing the type of the entry parameter: - String - Bool - MultipleChoice - Secret If you choose to define an entry parameter, this field is **required**. **Note:** This field cannot be modified on an edit.</td></tr><tr><td>RequiredWhen</td><td>An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: **HasPrivateKey**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. **OnAdd**: If set to *true*, a value must </td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>be provided for this field when configuring an add certificate job. The default is *false*. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. The default is *false*. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*.</td></tr><tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*. This value is unset by default.</td></tr><tr><td>Options</td><td>A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if *Type* is set to *MultipleChoice*. This value is unset by default.</td></tr></table> For example, to set a multiple choice entry parameter:

Name	In	Description

		<pre>"EntryParameter": [
		  {
		  "Name": "ZooAnimal",
		  "DisplayName": "Favorite Zoo Animal",
		  "Type": "MultipleChoice",
		  "RequiredWhen": {
		  "HasPrivateKey": false,
		  "OnAdd": true,
		  "OnRemove": true,
		  "OnReenrollment": true
		  },
		  "DefaultValue": "Penguin",
		  "Options": "Tiger-
		,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
		  }
		]</pre>
		This value is unset by default.
		
		Tip: What's the difference between properties (custom fields) and entry parameters?
		• Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record.
		• Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).

Table 359: POST Certificate Store Types Response Data

Name	Description
Name	A string containing the full name of the certificate store type.
ShortName	A string containing the short name assigned to the certificate store type.
Capability	A string containing a reference name for the certificate store type (e.g. NS for a NetScaler store).
StoreType	An integer indicating the Keyfactor Command reference ID for the certificate store type. The ID is automatically assigned by Keyfactor Command.
ImportType	An integer that indicates the import type for the certificate store type. The ID is automatically assigned by Keyfactor Command and generally matches the *StoreType* for custom certificate store types.
LocalStore	A Boolean that indicates whether the store is local to the orchestrator machine (true) as, for example, JKS and PEM stores managed by the Keyfactor Java Agent or remote (false) as, for example, IIS stores managed by the Keyfactor Universal Orchestrator.
SupportedOperations	An object containing Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of objects indicating unique parameters for the certificate store type. In the Keyfactor Command Management Portal these are known as *Custom Fields*. Property parameters include:

Name	Description
Name	A string containing the short name of the property.
DisplayName	A string containing the full display name of the property.
Type	A string containing the type of the property: • String • Bool

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• MultipleChoice
• Secret</td></tr>
<tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value(s) of the parameter. If Type is Multiple Choice, this field should contain a comma-separated list of multiple choice options for this parameter. If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that indicates whether the parameter is required (true) or not (false).</td></tr>
</table>

Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):
• ServerUsername
• ServerPassword
• ServerUseSsl

These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
</td></tr>
<tr><td>PasswordOptions</td><td>An object indicating options for the password in the certificate store type. Password options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EntrySupported</td><td>A Boolean that indicates whether entry of a password for the certificate in the certificate store is</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>allowed (true) or not (false).</td></tr>
<tr><td>StoreRequired</td><td>A Boolean that indicates whether entry of a password on the certificate store as a whole is required (true) or not (false).</td></tr>
<tr><td>Style</td><td>A string containing the style of password:
Default: Keyfactor Command will randomly generate a password.
Custom: Allow a password to be entered and authenticated when enrolling for a certificate through Keyfactor Command when installing to a store of this type. The Custom option can be selected only if the Allow Custom Password in the Application Settings, is equal to True. For more details, see Application Settings: Enrollment Tab on page 609.</td></tr>
</table>
</td></tr>
<tr><td>StorePathValue</td><td>A string containing the value(s) for the certificate store path.</td></tr>
<tr><td>PrivateKeyAllowed</td><td>A string containing the option for private key requirements for certificates stored in stores with this certificate store type:
Forbidden: Private key is not required; generally, applies to trust stores (e.g. Root CA certificates).
Optional: Private key is optional; applies to store types that could represent either a Trust Store or End-Entity Store.
Required: Private key is required; applies to stores that hold an End-Entity Certificate (server or client authorization).</td></tr>
<tr><td>ServerRequired</td><td>A Boolean that indicates whether server access is required for adding certificate stores for this certificate store type (true) or not (false). If set to true, a user will be prompted for a username and password to connect to the remote server.</td></tr>
<tr><td>PowerShell</td><td>A Boolean that indicates whether jobs for the store type are implemented using PowerShell (true) instead of a .NET class or not (false).</td></tr>
<tr><td>BlueprintAllowed</td><td>A Boolean that indicates whether certificate stores of this type will be included when creating or applying blueprints. For more details, see Orchestrator Blueprints on page 521.</td></tr>
<tr><td>CustomAliasAllowed</td><td>A string containing the selected certificate store type alias option:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Forbidden: A custom alias is not required and cannot be supplied.
Optional: A custom alias is optional.
Required: A custom alias is required.

The certificate store alias serves as an identifier for the certificate in the store. Depending on the type of store, it may be a file name, a certificate thumbprint, a string reference, or some other information. Some types of stores may not support associating an alias with the certificate (e.g. IIS trusted root).
</td></tr>
<tr><td>EntryParameters</td><td>
An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr>
<tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr>
<tr><td>Type</td><td>A string containing the type of the entry parameter:

String
Bool
MultipleChoice
Secret

</td></tr>
<tr><td>RequiredWhen</td><td>An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are:

HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store.
OnAdd: If set to true, a value must be provided for this field when configuring an add certi-

</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>ficate job. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job.</td></tr><tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*.</td></tr><tr><td>Options</td><td>A string containing a comma-separated list of multiple choice options for this entry parameter.</td></tr></table> **Tip:** What's the difference between properties (custom fields) and entry parameters? • Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record. • Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers

Name	Description
	with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).
InventoryEndpoint	A string containing the orchestrator endpoint to which inventory updates are sent.
InventoryJobType	A GUID identifying the job type for inventory jobs.
ManagementJobType	A GUID identifying the job type for management jobs.
DiscoveryJobType	A GUID identifying the job type for discovery jobs.
EnrollmentJobType	A GUID identifying the job type for reenrollment jobs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.12.7 PUT Certificate Store Types

The PUT /CertificateStoreTypes method is used to update a certificate store type in Keyfactor Command. This method returns HTTP 200 OK on a success with a message body containing a list of certificate store type details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Note: Certificate store types can only be updated in a very limited way if they are actively in use (there are any certificate stores defined for them). Updates to the Name, ShortName and adding a job type are supported in this case as are additions to the SupportedOperations, but no other updates can be saved.

Table 360: PUT Certificate Store Types Input Parameters

Name	In	Description
StoreType	Body	**Required**. An integer indicating the Keyfactor Command reference ID for the certificate store type.
Name	Body	**Required**. A string containing the full name of the certificate store type. A unique value must be supplied.
ShortName	Body	**Required**. A string containing the short name assigned to the certificate store type. A unique value must be supplied with a maximum of 10 characters.
Capability	Body	A string containing a reference name for the certificate store type (e.g. NS for a NetScaler store). **Note:** The *Capability* cannot be changed on an edit if an orchestrator has registered with Keyfactor Command, been approved, and included the certificate store type in its capability list.
LocalStore	Body	A Boolean that indicates whether the store is local to the orchestrator machine (true) as, for example, JKS and PEM stores managed by the Keyfactor Java Agent or remote (false) as, for example, IIS stores managed by the Keyfactor Universal Orchestrator. The default is *false*.
SupportedOperations	Body	An object containing Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove The default for each value is *false*.
Properties	Body	An array of objects indicating unique parameters for the certificate store type. In the Keyfactor Command Management Portal these are known as *Custom Fields*. Property parameters include: Name: StoreTypeID — Description: **Required**. An integer identifying the

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>certificate store type. This is the same ID referenced by the StoreType parameter, above. If you are updating a certificate store type, this field is **required**.</td></tr><tr><td>Name</td><td>**Required**. A string containing the short name of the property. If you choose to define a property, this field is **required**.</td></tr><tr><td>DisplayName</td><td>**Required**. A string containing the full display name of the property. If you choose to define a property, this field is **required**.</td></tr><tr><td>Type</td><td>**Required**. A string containing the type of the property: - String - Bool - MultipleChoice - Secret If you choose to define a property, this field is **required**. **Note:** This field cannot be modified on an edit.</td></tr><tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value(s) of the parameter. If *Type* is *Multiple Choice*, this field should contain a comma-separated list of multiple choice options for this parameter. If *Type* is</td></tr></table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that indicates whether the parameter is required (true) or not (false).</td></tr>
</table>
Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):

ServerUsername
ServerPassword
ServerUseSsl

These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
For example, to set a multiple choice property:
<pre>"Properties": [
   {
      "StoreTypeId": 111,
      "Name": "Pets",
      "DisplayName": "Popular Pets",
      "Type": "MultipleChoice",
      "DependsOn": "",
      "DefaultValue": "Cat,Dog,Fish,Rat,Mouse",
      "Required": false
   }
]</pre>
This value is unset by default.
</td></tr>
<tr><td>PasswordOptions</td><td>Body</td><td>An object indicating options for the password in the certificate store type. Password options include:</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EntrySupported</td><td>A Boolean that indicates whether entry of a password for the certificate in the certificate store is allowed (true) or not (false). The default is false.</td></tr>
<tr><td>StoreRequired</td><td>A Boolean that indicates whether entry of a password on the certificate store as a whole is required (true) or not (false). The default is false.</td></tr>
<tr><td>Style</td><td>A string containing the style of password:
Default: Keyfactor Command will randomly generate a password.
Custom: Allow a password to be entered and authenticated when enrolling for a certificate through Keyfactor Command when installing to a store of this type. The Custom option can be selected only if the Allow Custom Password in the Application Settings, is equal to True. For more details, see Application Settings: Enrollment Tab on page 609.
The default value is Default.</td></tr>
</table>
</td></tr>
<tr><td>StorePathType</td><td>Body</td><td>A string containing the selected store type:
Freeform: Users are required to enter a path defining the certificate store location.
Fixed: A store path does not apply, generally one store per device (e.g. IIS).
MultipleChoice: Allow a comma separated list of options to be entered that users will be able to select from when defining the certificate store location.
This value is unset by default.</td></tr>
</table>

Name	In	Description
StorePathValue	Body	A string containing the value(s) for the certificate store path if the *StorePathType* is set to Fixed or Multiple Choice. Multiple choice values should be provided in a bracketed comma-delimited list like so: `"StorePathValue": "[\"Apple\",\"Cherry\",\"Peach\",\"Pear\"]"` This value is unset by default.
PrivateKeyAllowed	Body	A string containing the option for private key requirements for certificates stored in stores with this certificate store type: • *Forbidden*: Private key is not required; generally, applies to trust stores (e.g. Root CA certificates). • *Optional*: Private key is optional; applies to store types that could represent either a Trust Store or End-Entity Store. • *Required*: Private key is required; applies to stores that hold an End-Entity Certificate (server or client authorization). The default value is *Forbidden*.
ServerRequired	Body	A Boolean that indicates whether server access is required for adding certificate stores for this certificate store type (true) or not (false). If set to true, a user will be prompted for a username and password to connect to the remote server. The default is *false*.
PowerShell	Body	A Boolean that indicates whether jobs for the store type are implemented using PowerShell (true) instead of a .NET class or not (false). The default is *false*.
BlueprintAllowed	Body	A Boolean that indicates whether certificate stores of this type will be included when creating or applying blueprints. For more details, see Orchestrator Blueprints on page 521. The default is *false*.
CustomAliasAllowed	Body	A string containing the selected certificate store type alias option: • *Forbidden*: A custom alias is not required and cannot be supplied. • *Optional*: A custom alias is optional. • *Required*: A custom alias is required. The certificate store alias serves as an identifier for the certificate in the store. Depending on the type of store, it may be a file

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>name, a certificate thumbprint, a string reference, or some other information. Some types of stores may not support associating an alias with the certificate (e.g. IIS trusted root).
The default value is Forbidden.</td></tr>
<tr><td>EntryParameters</td><td>Body</td><td>An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreTypeID</td><td>Required. An integer identifying the certificate store type. This is the same ID referenced by the StoreType parameter, above. If you are updating a certificate store type, this field is required.</td></tr>
<tr><td>Name</td><td>Required. A string containing the short name of the entry parameter. If you choose to define an entry parameter, this field is required. The name should be entered without spaces.</td></tr>
<tr><td>DisplayName</td><td>Required. A string containing the full display name of the entry parameter. If you choose to define an entry parameter, this field is required.</td></tr>
<tr><td>Type</td><td>Required. A string containing the type of the entry parameter:
String
Bool
MultipleChoice
Secret
If you choose to define an entry parameter, this field is required.
Note: This field cannot be modified on an edit.</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td>RequiredWhen</td><td>An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: **HasPrivateKey**: If set to *true*, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. The default is *false*. **OnAdd**: If set to *true*, a value must be provided for this field when configuring an add certificate job. The default is *false*. **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. The default is *false*. **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job. The default is *false*. </td></tr><tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value</td></tr></table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>for the entry parameter. If Type is Multiple Choice, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If Type is Boolean, this field should contain true or false.
This value is unset by default.</td></tr>
<tr><td>Options</td><td>A string containing a comma-separated list of multiple choice options for this entry parameter. This field should only be populated if Type is set to MultipleChoice.
This value is unset by default.</td></tr>
</table>
For example, to set a multiple choice entry parameter:
<pre>"EntryParameter": [
   {
      "StoreTypeId": 111,
      "Name": "ZooAnimal",
      "DisplayName": "Favorite Zoo Animal",
      "Type": "MultipleChoice",
      "RequiredWhen": {
         "HasPrivateKey": false,
         "OnAdd": true,
         "OnRemove": true,
         "OnReenrollment": true
      },
      "DefaultValue": "Penguin",
      "Options": "Tiger-
,Bear,Giraffe,Lion,Wolf,Penguin,Zebra"
   }
]</pre>
This value is unset by default.
Tip: What's the difference between properties (custom fields) and entry parameters?
Properties are about the certificate store definition itself and are static. For example, you might use a
</td></tr>
</table>

Name	In	Description
		property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record. • Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).

Table 361: PUT Certificate Store Types Response Data

Name	Description
Name	A string containing the full name of the certificate store type.
ShortName	A string containing the short name assigned to the certificate store type.
Capability	A string containing a reference name for the certificate store type (e.g. NS for a NetScaler store).
StoreType	An integer indicating the Keyfactor Command reference ID for the certificate store type. The ID is automatically assigned by Keyfactor Command.
ImportType	An integer that indicates the import type for the certificate store type. The ID is automatically assigned by Keyfactor Command and generally matches the *StoreType* for custom certificate store types.
LocalStore	A Boolean that indicates whether the store is local to the orchestrator machine (true) as, for example, JKS and PEM stores managed by the Keyfactor Java Agent or remote (false) as, for example, IIS stores managed by the Keyfactor Universal Orchestrator.
SupportedOperations	An object containing Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of objects indicating unique parameters for the certificate store type. In the Keyfactor Command Management Portal these are known as *Custom Fields*. Property parameters include:

Name	Description
Name	A string containing the short name of the property.
DisplayName	A string containing the full display name of the property.
Type	A string containing the type of the property: • String • Bool

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• MultipleChoice
• Secret</td></tr>
<tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value(s) of the parameter. If Type is Multiple Choice, this field should contain a comma-separated list of multiple choice options for this parameter. If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that indicates whether the parameter is required (true) or not (false).</td></tr>
</table>

Note: There are three standard properties that are used for certificate store types that require server credentials (e.g. F5):
• ServerUsername
• ServerPassword
• ServerUseSsl

These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.
</td></tr>
<tr><td>PasswordOptions</td><td>An object indicating options for the password in the certificate store type. Password options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EntrySupported</td><td>A Boolean that indicates whether entry of a password for the certificate in the certificate store is</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>allowed (true) or not (false).</td></tr>
<tr><td>StoreRequired</td><td>A Boolean that indicates whether entry of a password on the certificate store as a whole is required (true) or not (false).</td></tr>
<tr><td>Style</td><td>A string containing the style of password:
Default: Keyfactor Command will randomly generate a password.
Custom: Allow a password to be entered and authenticated when enrolling for a certificate through Keyfactor Command when installing to a store of this type. The Custom option can be selected only if the Allow Custom Password in the Application Settings, is equal to True. For more details, see Application Settings: Enrollment Tab on page 609.</td></tr>
</table>
</td></tr>
<tr><td>StorePathValue</td><td>A string containing the value(s) for the certificate store path.</td></tr>
<tr><td>PrivateKeyAllowed</td><td>A string containing the option for private key requirements for certificates stored in stores with this certificate store type:
Forbidden: Private key is not required; generally, applies to trust stores (e.g. Root CA certificates).
Optional: Private key is optional; applies to store types that could represent either a Trust Store or End-Entity Store.
Required: Private key is required; applies to stores that hold an End-Entity Certificate (server or client authorization).</td></tr>
<tr><td>ServerRequired</td><td>A Boolean that indicates whether server access is required for adding certificate stores for this certificate store type (true) or not (false). If set to true, a user will be prompted for a username and password to connect to the remote server.</td></tr>
<tr><td>PowerShell</td><td>A Boolean that indicates whether jobs for the store type are implemented using PowerShell (true) instead of a .NET class or not (false).</td></tr>
<tr><td>BlueprintAllowed</td><td>A Boolean that indicates whether certificate stores of this type will be included when creating or applying blueprints. For more details, see Orchestrator Blueprints on page 521.</td></tr>
<tr><td>CustomAliasAllowed</td><td>A string containing the selected certificate store type alias option:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

Forbidden: A custom alias is not required and cannot be supplied.
Optional: A custom alias is optional.
Required: A custom alias is required.

The certificate store alias serves as an identifier for the certificate in the store. Depending on the type of store, it may be a file name, a certificate thumbprint, a string reference, or some other information. Some types of stores may not support associating an alias with the certificate (e.g. IIS trusted root).
</td></tr>
<tr><td>EntryParameters</td><td>
An array of objects indicating unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr>
<tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr>
<tr><td>Type</td><td>A string containing the type of the entry parameter:

String
Bool
MultipleChoice
Secret

</td></tr>
<tr><td>RequiredWhen</td><td>An object containing Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are:

HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store.
OnAdd: If set to true, a value must be provided for this field when configuring an add certi-

</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>ficate job. • **OnRemove**: If set to *true*, a value must be provided for this field when configuring a remove certificate job. • **OnReenrollment**: If set to *true*, a value must be provided for this field when configuring a reenrollment job.</td></tr><tr><td>DependsOn</td><td>A string containing the name of the parameter on which this parameter depends. This only applies if at least two custom parameters have been created for this certificate store type. This option is used to configure one custom parameter to display only if another custom parameter contains a value.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value for the entry parameter. If *Type* is *Multiple Choice*, this field should contain a single value that represents the default selection from the provided list (see Options) for this entry parameter. If *Type* is *Boolean*, this field should contain *true* or *false*.</td></tr><tr><td>Options</td><td>A string containing a comma-separated list of multiple choice options for this entry parameter.</td></tr></table> **Tip:** What's the difference between properties (custom fields) and entry parameters? • Properties are about the certificate store definition itself and are static. For example, you might use a property to define the primary node name of an F5 instance. This node name is the same no matter what inventory or management jobs you do with the F5 device(s). Values for properties are entered in the certificate store record when creating or editing the certificate store record. • Entry parameters are about the specific certificate within the certificate store. They are used to send additional information related to the certificate to the server or device that hosts the certificate store when running management jobs for that certificate store. Often this is more fluid information that isn't the same for every use of that certificate store. For example, several virtual servers

Name	Description
	with separate certificates in the same folder may exist on a NetScaler device. When replacing one certificate, updates may need to be made to only the virtual server that is using the certificate. In this case, the authorized user will be prompted to enter the virtual server name based on an entry parameter. Values for entry parameters are entered at the time a management job is initiated (e.g. adding a certificate to a certificate store).
InventoryEndpoint	A string containing the orchestrator endpoint to which inventory updates are sent.
InventoryJobType	A GUID identifying the job type for inventory jobs.
ManagementJobType	A GUID identifying the job type for management jobs.
DiscoveryJobType	A GUID identifying the job type for discovery jobs.
EnrollmentJobType	A GUID identifying the job type for reenrollment jobs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.13 Component Installation

The Component Installation methods are used to list the servers that various Keyfactor Command components have been installed on or to decommission a Keyfactor Command server that is no longer in use.

Table 362: Component Installation Endpoints

Endpoint	Method	Description	Link
/	GET	Return a list of all the Keyfactor Command servers in the current implementation including the components installed on each, with paging (number of pages to return and number of results per page) and sorting options.	GET Component Installation on the next page
/{id}	DELETE	Delete the Keyfactor Command server with the	DELETE

Endpoint	Method	Description	Link
		specified ID from the Keyfactor Command database.	Component Installation ID on the next page

3.6.13.1 DELETE Component Installation ID

The DELETE /ComponentInstallation/{id} method is used to delete the Keyfactor Command server with the specified ID and all its components from the Keyfactor Command database. This endpoint returns 204 with no content upon success.

Important: Servers should not be deleted if they are serving any active role in the Keyfactor Command environment, as this operation cannot be undone.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/system_settings/modify/

Table 363: DELETE Component Installation {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the Keyfactor Command server to delete. Use the *GET /ComponentInstallation* method (see GET Component Installation below) to determine the ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.13.2 GET Component Installation

The GET /ComponentInstallation method is used to return a list of all the Keyfactor Command components installed for the current implementation. This method returns HTTP 200 OK on a success with details about the installed components on each server. This method allows URL parameters to specify paging and the level of information detail.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/system_settings/read/

Table 364: GET Component Installation Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • Machine • Version
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 365: GET Component Installation Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the server installation.
Machine	A string containing the FQDN of the server.
Version	A string containing the version of Keyfactor Command installed on the server.
Components	A string containing a comma-separated list of the components installed on the server. Possible components include: • Console The server with this role provides the web-based administration interface (the Management Portal) that is used to view and report on certificates issued in the environment and enroll for certificates. This role is required on all Keyfactor Command servers. • Logi The server with this role hosts the Logi Analytics Platform for reporting. This role is required on all Keyfactor Command servers. • Agents The server with this role hosts the back-end service for receiving requests from and sending requests to Keyfactor agents and orchestrators. This role is optional. • KeyfactorAPI The server with this role hosts the newer Keyfactor API. This role is required on all Keyfactor Command servers. • Service The server with this role hosts back-end services required to support Keyfactor Command. This includes the Keyfactor Command Service, which is used for all periodic tasks throughout Keyfactor Command, including CA synchronization, monitoring alerts, and report automation. This role is required on all Keyfactor Command servers.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.14 CSR Generation

The CSR Generation component of the Keyfactor API includes methods necessary to generate certificate signing requests and determine which ones are pending.

Table 366: CSR Generation Endpoints

Endpoint	Method	Description	Link
/Pending/{id}	DELETE	Deletes a pending CSR by ID.	DELETE CSR Generation Pending ID below
/Pending/{id}	GET	Returns the details of a specific CSR request based on the ID number.	GET CSR Generation Pending ID on the next page
/Pending	DELETE	Deletes multiple pending CSRs.	DELETE CSR Generation Pending on the next page
/Pending	GET	Returns a list of all pending CSRs.	GET CSR Generation Pending on page 1588
/Generate	POST	Generate and configure a CSR request.	POST CSR Generation Generate on page 1589

3.6.14.1 DELETE CSR Generation Pending ID

The DELETE /CSRGeneration/Pending/{id} method is used to delete a certificate signing request with the defined ID that has not yet been enrolled. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/requests/manage/

Table 367: DELETE CSR Generation Pending {id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the certificate signing request for the CSR that should be deleted. Use the *GET /CSRGeneration/Pending* method (see GET CSR Generation Pending on page 1588) to retrieve a list of all the pending CSRs to determine the CSR IDs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.14.2 GET CSR Generation Pending ID

The GET /CSRGeneration/Pending/{id} method is used to return a generated CSR with the defined ID that has not yet been enrolled. This method returns HTTP 200 OK on a success with the CSR in PEM format. This method does not return the parsed subject name or CSR request time. If you need that information, use the *GET /CSRGeneration/Pending* method (see GET CSR Generation Pending on the next page).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/requests/manage/

Table 368: GET CSR Generation Pending {id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the CSR that should be retrieved.

Table 369: GET CSR Generation Pending {id} Response Data

Name	Description
CSRFilePath	The proposed file name for the CSR file. This is considered deprecated and may be removed in a future release.
CSR	The text of the CSR in PEM format.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.14.3 DELETE CSR Generation Pending

The DELETE /CSRGeneration/Pending method is used to delete multiple certificate signing requests that have not yet been enrolled in one request. Delete operations will continue until the entire array of IDs has been processed. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/requests/manage/

Table 370: DELETE CSR Generation Pending Input Parameters

Name	In	Description
ids	Body	**Required**. An array of integers indicating the Keyfactor Command certificate signing request IDs for CSRs that should be deleted in the form (without parameter name): `[8,14,27]` Use the *GET /CSRGeneration/Pending* method (see GET CSR Generation Pending below) to retrieve a list of all the pending CSRs to determine the CSR IDs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.14.4 GET CSR Generation Pending

The GET /CSRGeneration/Pending method is used to return details for generated CSRs that have not yet been enrolled. This method returns HTTP 200 OK on a success with details of the pending CSRs with details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/requests/manage/

Table 371: GET CSR Generation Pending Input Parameters

Name	In	Description
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 372: GET CSR Generation Pending Response Data

Name	Description
Id	A unique integer for the CSR generated.
CSR	A string containing the text of the CSR in PEM format.
RequestTime	A string containing the date and time that the CSR was generated in UTC time.
Subject	An array or strings containing the subject of the certificate including the certificate subject information, the subject alternative names, the key length, and the hash algorithm.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.14.5 POST CSR Generation Generate

The POST /CSRGeneration/Generate method is used to generate and configure a CSR. This method returns HTTP 200 OK on a success with a message body containing the text of the CSR file created.

This method generates a private key and stores it in the Keyfactor Command database. When you use the CSR resulting from this method to enroll for a certificate through Keyfactor Command (see POST Enrollment CSR on page 1658), the resulting certificate is married together with the stored private key and may then be download with private key (see POST Certificates Recover on page 1169).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/enrollment/csr/generation/

Note: This endpoint no longer includes the CSRFilePath return value in the response from the API call. Code separate from the API should be used to handle receipt of the CSR and placement on the file system.

Table 373: POST CSR Generation Generate Input Parameters

Name	In	Description
Curve	Body	A string indicating the elliptic curve for the requested key. ECC curves may be specified using the well-known OIDs for ECC algorithms. Well-known OIDs include: • 1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1 • 1.3.132.0.34 = P-384/secp384r1 • 1.3.132.0.35 = P-521/secp521r1
KeyLength	Body	**Required***. An integer indicating the desired key size of the certificate. Supported key sizes are: • 255 • 256 • 384 • 448 • 521 • 2048 • 3072 • 4096 • 8192 This value is **required** only if *KeyType* = RSA.
KeyType	Body	**Required**. A string indicating the desired key encryption of the certificate. Supported key types are: • RSA • ECC • Ed448 • Ed25519
SANs	Body	An object that contains the elements for Keyfactor Command to use when generating the subject alternative name (SAN) for the certificate requested by the CSR, each of which is supplied as an array of strings. Possible values for the key are:

Value	Description
rfc822	RFC 822 Name
dns	DNS Name

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>directory</td><td>Directory Name</td></tr>
<tr><td>uri</td><td>Uniform Resource Identifier</td></tr>
<tr><td>ip4</td><td>IP v4 Address</td></tr>
<tr><td>ip6</td><td>IP v6 Address</td></tr>
<tr><td>registeredid</td><td>Registered ID (an OID)</td></tr>
<tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr>
<tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr>
</table>
For example:
<pre>"SANs": {
   "dns": [
      "dnssan1.keyexample.com",
      "dnssan2.keyexample.com",
      "dnssan3.keyexample.com"
   ],
   "ip4": [
      "192.168.2.73"
   ]
}</pre>
</td></tr>
<tr><td>Subject</td><td>Body</td><td>
Required. A string containing the subject name for the certificate using X.500 format for the full distinguished name (DN). For example:
<pre>"Subject": "CN=websrvr14.keyexample.com,OU=IT,O=\"Key Example, Inc.\",L=Independence,ST=OH,C=US"</pre>
Supported subject name fields are:
<table>
<tr><th>Name</th><th>Abbreviation</th><th>Description</th></tr>
<tr><td>CommonName</td><td>CN</td><td>Required*. The desired common name of the certificate to be requested with the</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Abbreviation</th><th>Description</th></tr><tr><td></td><td></td><td>CSR. This field is **required** if the *Common Name Regular Expression* application setting is set to the default value of *.+*. See Application Settings: Enrollment Tab on page 609 for more information.</td></tr><tr><td>Organization</td><td>O</td><td>The desired organization of the certificate to be requested with the CSR.</td></tr><tr><td>OrganizationalUnit</td><td>OU</td><td>The desired organizational unit of the certificate to be requested with the CSR.</td></tr><tr><td>Locality</td><td>L</td><td>The desired city of the certificate to be requested with the CSR.</td></tr><tr><td>State</td><td>ST</td><td>The desired state of the certificate to be requested with the CSR.</td></tr><tr><td>Country</td><td>C</td><td>The desired country (two characters) of the certificate to be requested with the CSR.</td></tr><tr><td>Email</td><td>E</td><td>The desired email address of the certificate to be requested with the CSR.</td></tr></table>

Name	In	Description
Template	Body	A string indicating the desired template to be used for the certificate to be requested with the CSR. The template must have been configured in Keyfactor Command to support CSR generation. This field is optional. **Important:** The template will not be included in the CSR. The template is referenced in order to retrieve key and other information to help populate the CSR. In addition, the CSR generation function supports template-level regular expressions for both subject parts and SANs. If system-wide and template-level regular expressions exists for the same field and you select a template, the template-level regular expression is applied. If you choose to select a template during CSR generation, you will need to choose the same template during CSR Enrollment, because the CSR file will contain elements from the template which may conflict with other template configurations.

Table 374: POST CSR Generation Generate Response Data

Name	Description
CSR	The text of the CSR in PEM format.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.15 Custom Job Types

The Custom Job Types component of the Keyfactor API includes methods necessary to create, update, list and delete custom orchestrator job types. Custom job types are intended to execute jobs on an orchestrator built using the AnyAgent framework that are outside the standard list of job functions built into Keyfactor Command. This powerful feature can execute just about any job that requires processing on the orchestrator and submitting data back to Keyfactor Command. The data submitted by custom jobs to Keyfactor Command is stored as a string and is limited to 2 MB.

Table 375: Custom Job Types Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the custom job type for the specified ID.	DELETE Custom Job Types ID below
/{id}	GET	Returns details for the custom job type for the specified ID.	GET Custom Job Types ID on the next page
/	GET	Returns all the custom job types.	GET Custom Job Types on page 1598
/	POST	Creates a custom job type.	POST Custom Job Types on page 1600
/	PUT	Updates an existing custom job type.	PUT Custom Job Types on page 1604

3.6.15.1 DELETE Custom Job Types ID

The DELETE /JobTypes/Custom/{id} method is used to delete an existing custom job type with the specified GUID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/modify/

Table 376: DELETE JobTypes Custom {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference GUID of the custom job type. Use the *GET /JobTypes/Custom* method (see GET Custom Job Types on page 1598) to retrieve a list of all the custom job types to determine the job type GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.15.2 GET Custom Job Types ID

The GET /JobTypes/Custom/{id} method is used to return a custom job type with the specified GUID. This method returns HTTP 200 OK on a success with details for the custom job type.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 377: GET JobTypes Custom {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference GUID of the custom job type. Use the *GET /JobTypes/Custom* method (see GET Custom Job Types on page 1598) to retrieve a list of all the custom job types to determine the job type GUID.

Table 378: GET JobTypes Custom {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>JobTypeName</td><td>A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it.</td></tr>
<tr><td>Description</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>A value that indicates the data type of the job type field.
Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table>
</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false).</td></tr>
</table>
</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.15.3 GET Custom Job Types

The GET /JobTypes/Custom method is used to retrieve a list of all custom job types. This method returns HTTP 200 OK on a success with details for each job type.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 379: GET Job Types Custom Input Parameters

Name	In	Description
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.

Table 380: GET Job Types Custom Response Data

Name	Description
Id	The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.
JobTypeName	A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it.
Description	A string containing a description for the custom job type.
JobTypeFields	An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:

Name	Description
Name	A string that indicates the name for the job type field.
Type	A value that indicates the data type of the job type field. Possible values are:

Integer Value	Enum Value	Description
1	String	String
2	Int	Integer
3	DateTime	Date
4	Bool	Boolean

DefaultValue	A string containing the default value of the job type field. If *Type* is *Boolean*, this field should contain *true* or *false*.
Required	A Boolean that sets whether the job type field is required (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.15.4 POST Custom Job Types

The POST /JobTypes/Custom method is used to create a custom orchestrator job type in Keyfactor Command. This method returns HTTP 200 OK on a success with a message body containing a list of custom job type details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/modify/

Table 381: POST JobTypes Custom Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>JobTypeName</td><td>Body</td><td>Required. A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it. This name should not contain spaces.</td></tr>
<tr><td>Description</td><td>Body</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>Body</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>Required. A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>Required. A value that indicates the data type of the job type field.
It may be entered as either an integer or the matching enum value. Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table></td></tr>
<tr><td>DefaultValue</td><td>Required*. A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.
This field is required if the Required parameter is set to true.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false). The default is false.</td></tr>
</table>
For example:</td></tr>
</table>

Name	In	Description

		"JobTypeFields": [
		{
		"Name": "Favorite Type of Pet",
		"Type": "String",
		"DefaultValue": "Cat",
		"Required": true
		},
		{
		"Name": "Model Year of First Car",
		"Type": "Int"
		},
		{
		"Name": "Mother's Birthday",
		"Type": "DateTime"
		}
		]

Table 382: POST JobTypes Custom Response Data

Name	Description
Id	The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.
JobTypeName	A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it.
Description	A string containing a description for the custom job type.
JobTypeFields	An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:

Name	Description
Name	A string that indicates the name for the job type field.
Type	A value that indicates the data type of the job type field. Possible values are:

Integer Value	Enum Value	Description
1	String	String
2	Int	Integer
3	DateTime	Date
4	Bool	Boolean

DefaultValue	A string containing the default value of the job type field. If *Type* is *Boolean*, this field should contain *true* or *false*.
Required	A Boolean that sets whether the job type field is required (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.15.5 PUT Custom Job Types

The PUT /JobTypes/Custom method is used to create a custom orchestrator job type in Keyfactor Command. This method returns HTTP 200 OK on a success with a message body containing a list of certificate store type details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 383: PUT JobTypes Custom Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td>Body</td><td>The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>JobTypeName</td><td>Body</td><td>Required. A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it. This name should not contain spaces.</td></tr>
<tr><td>Description</td><td>Body</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>Body</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>Required. A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>Required. A value that indicates the data type of the job type field.
It may be entered as either an integer or the matching enum value. Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table></td></tr>
<tr><td>DefaultValue</td><td>Required*. A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.
This field is required if the Required parameter is set to true.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false). The default is false.</td></tr>
</table></td></tr>
</table>

Name	In	Description

		For example:
		
		"JobTypeFields": [
		{
		"Name": "Favorite Type of Pet",
		"Type": "String",
		"DefaultValue": "Cat",
		"Required": true
		},
		{
		"Name": "Model Year of First Car",
		"Type": "Int"
		},
		{
		"Name": "Mother's Birthday",
		"Type": "DateTime"
		}
		]

Table 384: PUT JobTypes Custom Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>JobTypeName</td><td>A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it.</td></tr>
<tr><td>Description</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>A value that indicates the data type of the job type field.
Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table>
</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false).</td></tr>
</table>
</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16 Enrollment

The Enrollment component of the Keyfactor API includes methods necessary to enroll certificate signing requests (CSRs) and personal information exchanges (PFXs).

Table 385: Enrollment Endpoints

Endpoint	Method	Description	Link
/Settings/{Id}	GET	Returns the template settings to use during enrollment.	GET Enrollment Settings ID on the next page
/CSR/Context/My	GET	Returns the templates available for CSR enrollment by the current user.	GET Enrollment CSR Content My on page 1616
/PFX/Context/My	GET	Returns the templates available for PFX enrollment by the current user.	GET Enrollment PFX Content My on page 1634
/AvailableRenewal/Id/{id}	GET	Returns the type of renewals available for the referenced certificate ID.	GET Enrollment Available Renewal ID on page 1653
/AvailableRenewal/Thumbprint/{thumbprint}	GET	Returns the type of renewals available for the referenced certificate thumbprint.	GET Enrollment Available Renewal Thumbprint on page 1656
/CSR	POST	Performs a CSR enrollment.	POST Enrollment CSR on page 1658
/PFX	POST	Performs a PFX enrollment.	POST Enrollment PFX on page 1665
/CSR/Parse	POST	Returns information found in a CSR in a human friendly form.	POST Enrollment CSR Parse on page 1683
/PFX/Deploy	POST	Adds a certificate into a certificate store following a PFX enrollment or certificate renewal.	POST Enrollment PFX Deploy on page 1685
/PFX/Replace	POST	Replaces a certificate in a certificate store following a PFX enrollment.	POST Enrollment PFX Replace on page 1691

Endpoint	Method	Description	Link
/Renew	POST	Performs a certificate renewal.	POST Enrollment Renew on page 1694

3.6.16.1 GET Enrollment Settings ID

The GET /Enrollment/Settings/{id} method is used to return the template settings to use during enrollment for a given template. The response will be the resolved values for the template settings (based on whether they are global or template-specific). This method returns HTTP 200 OK on a success with details of the template regular expressions, defaults, and policy. If there is a template-specific setting, the template-specific setting will be shown in the response. If there is not a template-specific setting, the global settings will be shown in the response.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/enrollment/csr/
OR
/certificates/enrollment/pfx/
OR
/certificates/enrollment/csr/generation/

Table 386: GET Enrollment Settings {id} Input Parameters

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the enrollment template. Use the *GET /Templates* method (see GET Templates on page 2422) to retrieve a list of all the templates to determine the template ID.

Table 387: GET Enrollment Settings {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateRegexes</td><td>An array of objects containing the regular expressions resolved for the template. Regular expression details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see <a>GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly .keyexample.com:
<pre>^[a-zA-Z0-9'_\.\-]*\.keyexample\.com$</pre>The default value for the Common Name regular expression is:
<pre>.+</pre>This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Name	Description
	Name / **Description**

Subject Part	Example
O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA:

Name	Description
	Name / **Description**

Subject Part	Example
	`^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$`

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Subject Part</th><th>Example</th></tr><tr><td></td><td>This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods: <code>^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$</code></td></tr><tr><td>IPv6 (Subject Alternative Name: IPv6 Address)</td><td>This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: <code>^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$</code></td></tr><tr><td>MAIL (Subject Alternative Name: Email)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: <code>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</code></td></tr><tr><td>UPN (Subject Alternative Name: User Principal Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: <code>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</code></td></tr></table></td></tr><tr><td>Error</td><td>A string specifying the error message displayed to the user when</td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.

Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
</td></tr>
<tr><td>TemplateDefaults</td><td>An array of objects containing the template defaults resolved for the template. Template-level defaults, if defined, take precedence over global-level template defaults. For more information about global-level template defaults, see <a>GET Templates Settings on page 2395. The template default object contains the following parameters:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the default applies to (e.g. L for City/Locality).
Use the GET /Templates/SubjectParts method (see <a>GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.</td></tr>
<tr><td>Value</td><td>A string containing the value to assign as the default for that subject part (e.g. Chicago).</td></tr>
</table>
</td></tr>
<tr><td>TemplatePolicy</td><td>An object containing the template policy settings. The template policy object contains the following parameters:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>AllowKeyReuse</td><td>A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option allows to certificate renewals.</td></tr>
<tr><td>AllowWildcards</td><td>A Boolean that indicates whether wildcards are allowed (true) or not (false).</td></tr>
<tr><td>RFCEnforcement</td><td>A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this</td></tr>
</table>
</td></tr>
</table>

Name	Description

Value	Description
	option is set to *true*, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor CommandManagement Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set.
KeyInfo	An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate. Key info includes:

Name	Description
ECDSA	An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command.
RSA	An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no curves for this type of key.
Ed448	An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed25519</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.2 GET Enrollment CSR Content My

The GET /Enrollment/CSR/Context/My method is used to check the templates and CAs available for CSR enrollment for the current user. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849). It returns HTTP 200 OK on a success with the list of templates that are available for enrollment via Keyfactor Command and the CAs those templates may be enrolled from along with template and CA configuration details. Results are returned based on the enrollment permissions of the user making the request—both Keyfactor Command permissions and template and CA level permissions on the originating CA. Templates or standalone CAs are included in the results only if the user has appropriate permissions in both locations and the template and CA are configured for CSR enrollment in Keyfactor Command.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/enrollment/csr/

Table 388: GET Enrollment CSR Content My Response Data

Name	Description
Templates	An array of objects containing the templates available for enrollment by the user. Each object contains the following parameters:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the certificate template.
Name	A string containing the common name (short name) of the template. This name typically does not contain spaces. For a template created using a Microsoft management tool, this will be the Microsoft template name.
DisplayName	A string containing the common name (short name) of the template. This name typically does not contain spaces. For a template created using a Microsoft management tool, this will be the Microsoft template name.
RequiresApproval	A Boolean indicating whether the template has been configured with the Microsoft *CA certificate manager approval* option enabled (true) or not (false).
RFCEnforcement	A Boolean indicating whether certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN (true) or not (false). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to *false* at a system-wide level and may be overridden on a template-by-template basis.
CAs	An array of objects indicating the certificate authorities that allow enrollment for the template and the requesting user. The template must be available for enrollment on the CA, the template and CA must be configured for enrollment in Keyfactor Command, and the requesting user must have enrollment permissions. Information about the CA includes:

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Name: A string indicating the full name of the CA, made up of the DNS hostname of the certificate authority (e.g. corpca01.keyexample.com) and the logical name (e.g. CorpIssuingCA1) for a full name similar to corpca01.keyexample.com\\CorpIssuingCA1.
	RFCEnforcement: A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (*true*) or not (*false*). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. This setting at the CA level applies only to standalone CAs. For CAs that use templates, this setting is controlled at the template level and is ignored at the CA level.
	SubscriberTerms: A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (*true*) or not (*false*). **Tip:** Configure a link to the custom terms using the *URL to Subscriber Terms* application setting. See Application Settings: Enrollment Tab on page 609 for more information.

Name	Description

Name	Description
Enroll-mentFields	An array of objects containing custom enrollment fields. These are configured on a per-template basis to allow you to submit custom fields with CSR enrollments and PFX enrollments to supply custom request attributes to the CA during the enrollment process. This functionality offers such benefits as: • Preventing users from requesting invalid certificates, based on your specific certificate requirements per template. • Providing additional information to the CA with the CSR. Once created on the template, these values are shown in Keyfactor Command on the PFX and CSR enrollment pages in the *Additional Enrollment Fields* section. The fields are mandatory during enroll-ment. The data will appear on the CA / Issued Certificates attribute tab for certificates enrolled with a template configured with Keyfactor Command enrollment fields. **Note:** These are not metadata fields, so they are not stored in the Keyfactor Command database, but simply passed through to the CA. The CA in turn could, via a gateway or policy module, use this data to perform required actions. The array contains the following parameters:

Name	Description
Id	An integer indicating the ID of the custom enrollment field.
Name	A string indicating the name of the custom enrollment field. This name will appear on the enrollment pages.
Options	For multiple choice values, an array of strings containing the value choices.
DataType	An integer indicating the parameter type. The options are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>String: A free-form data entry field.</td></tr>
<tr><td>2</td><td>Multiple Choice: Provides a list of acceptable values for the field. The multiple choice values are provided in the Options parameter.</td></tr>
</table>
For example:
<pre>
"EnrollmentFields": [
   {
      "Id": 3,
      "Name": "MyCustomField",
      "Options": ["Green","Red","Yellow","Blue"],
      "DataType": 2
   }
]
</pre>
</td></tr>
<tr><td>MetadataFields</td><td>An array of objects containing template-level metadata field settings. Template-level metadata field configurations can override global metadata field configurations in these possible ways:
Configuration on the metadata field of required, optional or hidden.
The default value for the metadata field.
A regular expression defined for the field (string fields only) against which entered data will be validated along with its associated message.
For fields of data type multiple choice, the list of values that appear in multiple choice dropdowns.
Metadata field settings defined on a template apply to enrollments made with that template only. Template-level metadata field settings, if defined, take precedence over global-level metadata</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**
	field settings. The metadata field settings array contains the following parameters:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the template-specific metadata setting.
DefaultValue	A string containing the default value defined for the metadata field for the specific template.
MetadataId	An integer indicating the global metadata field associated with the template-specific settings.
Validation	A string containing the template-specific regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the *Message* field. For example: `^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$` This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”. This field is only supported for metadata

Name	Description
	Name / **Description**
	fields with data type *string*.
Enrollment	An integer that indicates how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are: **Value** / **Description** 0 — Optional Users have the option to either enter a value or not enter a value in the field. 1 — Required Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page. 2 — Hidden The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.
Message	A string containing a message to present when a user enters information in a metadata field that does not match the template-specific regular expression (*Validation* field).

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example:
<pre>"MetadataFields": [
   {
      "Id": 4,
      "DefaultValue": "reggie.wallace@keyexample.com",
      "MetadataId": 4,
      "Validation": "^[a-zA-Z0-9'_\\.\\-]*@(keyexample\\.or-
g|keyexample\\.com)$",
      "Enrollment": 1,
      "Message": "Your email address must be of the form
user@keyexample.com or fname.lname@keyexample.com."
   },
   {
      "Id": 13,
      "DefaultValue": "E-Business",
      "MetadataId": 5,
      "Validation": "",
      "Enrollment": 0,
      "Message": "",
      "Options": "Accounting,E-Busi-
ness,Executive,HR,IT,Marketing,R&D,Sales"
   }
]</pre></td></tr>
<tr><td>Regexes</td><td>An array of objects containing the global template regular expression settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Regular expression details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>In integer indicating the Keyfactor Command reference ID of the certificate template the regular expression is associated with.</td></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against</td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**
	Name / **Description**
	which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated. Use the *GET /Templates/SubjectParts* method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts. The following are some regular expression examples:

Subject Part	Example

CN (Common Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Subject Part / **Example**
	one character in the Common Name field in the enrollment pages.
O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities:

Name	Description

Name	Description

Name	Description

Subject Part	Example
	`^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_`

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Subject Part / **Example**
	`\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$`

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Subject Part / **Example**
	This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods: `^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`
	IPv6 (Subject Alternative Name: IPv6 Address) — This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
	MAIL (Subject Alternative Name: Email) — This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>UPN (Subject Alternative Name: User Principal Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:

<code>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</code></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.

Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
</td></tr>
<tr><td>ExtendedKeyUsages</td><td>Currently not in use.</td></tr>
<tr><td>EnrollmentTemplatePolicy</td><td>An object containing the individual template-level template policy settings. Template policies defined on a template apply to enrollments made with that template only. Template-level policies, if defined, take precedence over system-wide template policies. For</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description** more information about system-wide template policies, see GET Templates Settings on page 2395. The template policy object contains the following parameters: **Value** / **Description** KeyInfo — An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate. **Name** / **Description** ECDSA — An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command. ECC curves may be specified using the well-known OIDs for ECC algorithms or by friendly name. Well-known OIDs include: • 1.2.840.10045.3.1.7 = P-

Name	Description
	Name / **Description** **Value** / **Description** **Name** / **Description** 256/prime256v1/secp256r1 • 1.3.132.0.34 = P-384/secp384r1 • 1.3.132.0.35 = P-521/secp521r1 When specifying by friendly name, do not include a slash (use “P-256”, not “P-256/prime256v1/secp256r1”). RSA: An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no curves for this type of key. Ed448: An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no

Name	Description
	Name / **Description** **Value** / **Description** **Name** / **Description** curves for this type of key. Ed25519 — An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no curves for this type of key. AllowKeyReuse — A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option applies to certificate renewals. By default, this is set to *true* at a system-wide level. AllowWildcards — A Boolean that indicates whether wildcards are allowed (true) or not (false). By default, this is set to *true* at a system-wide level. RFCEnforcement — A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to *true*, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to false at a system-wide level.</td></tr>
</table>
For example:
<pre>
"TemplatePolicy": {
   "AllowKeyReuse": false,
   "AllowWildcards": true,
   "RFCEnforcement": true,
   "KeyInfo": {
      "ECDSA": {
         "bit_lengths": [
            256,
            384,
            521
         ],
         "curves": [
            "1.2.840.10045.3.1.7",
            "1.3.132.0.34",
            "1.3.132.0.35"
         ]
      },
      "RSA": null,
      "Ed448": null,
      "Ed25519": null
   }
}
</pre>
</td></tr>
<tr><td>KeySize</td><td>A string indicating the minimum supported key size of the template.</td></tr>
<tr><td>Curve</td><td>A string indicating the OID of the elliptical curve algorithm configured for the template, for ECC templates.</td></tr>
</table>
</td></tr>
<tr><td>StandaloneCAs</td><td>An array of objects containing enrollment information for standalone certificate authorities available for enrollment for the current user. Information about the CA includes:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Name</td><td>The full name of the CA, made up of the DNS hostname of the certificate authority (e.g. myca.keyexample.com) and the logical name (e.g. CorpStandaloneCA1) for a full name similar to myca.keyexample.com\\CorpStandaloneCA1.</td></tr><tr><td>RFCEnforcement</td><td>A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (true) or not (false). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. This setting at the CA level applies only to standalone CAs. For CAs that use templates, this setting is controlled at the template level and is ignored at the CA level.</td></tr><tr><td>SubscriberTerms</td><td>A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (true) or not (false). Tip: Configure a link to the custom terms using the URL to Subscriber Terms application setting. See Application Settings: Enrollment Tab on page 609 for more information.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.3 GET Enrollment PFX Content My

The GET /Enrollment/PFX/Context/My method is used to check the templates and CAs available for PFX enrollment for the current user. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849). It returns HTTP 200 OK on a success with the list of templates that are available for enrollment via Keyfactor Command and the CAs those templates may be enrolled from along with template and CA configuration details. Results are returned based on the enrollment permissions of the user making the request—both Keyfactor

Command permissions and template and CA level permissions on the originating CA. Templates or standalone CAs are included in the results only if the user has appropriate permissions in both locations and the template and CA are configured for PFX enrollment in Keyfactor Command.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/enrollment/pfx/

Table 389: GET Enrollment PFX Content My Response Data

Name	Description
Templates	An array of objects containing the templates available for enrollment by the user. Each object contains the following parameters:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the certificate template.
Name	A string containing the common name (short name) of the template. This name typically does not contain spaces. For a template created using a Microsoft management tool, this will be the Microsoft template name.
DisplayName	A string containing the common name (short name) of the template. This name typically does not contain spaces. For a template created using a Microsoft management tool, this will be the Microsoft template name.
RequiresApproval	A Boolean indicating whether the template has been configured with the Microsoft *CA certificate manager approval* option enabled (true) or not (false).
RFCEnforcement	A Boolean indicating whether certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN (true) or not (false). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to *false* at a system-wide level and may be overridden on a template-by-template basis.
CAs	An array of objects indicating the certificate authorities that allow enrollment for the template and the requesting user. The template must be available for enrollment on the CA, the template and CA must be configured for enrollment in Keyfactor Command, and the requesting user must have enrollment permissions. Information about the CA includes:

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Name: A string indicating the full name of the CA, made up of the DNS hostname of the certificate authority (e.g. corpca01.keyexample.com) and the logical name (e.g. CorpIssuingCA1) for a full name similar to corpca01.keyexample.com\\CorpIssuingCA1.
	RFCEnforcement: A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (*true*) or not (*false*). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. This setting at the CA level applies only to standalone CAs. For CAs that use templates, this setting is controlled at the template level and is ignored at the CA level.
	SubscriberTerms: A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (*true*) or not (*false*). **Tip:** Configure a link to the custom terms using the *URL to Subscriber Terms* application setting. See Application Settings: Enrollment Tab on page 609 for more information.

Name	Description

Name	Description
Enroll-mentFields	An array of objects containing custom enrollment fields. These are configured on a per-template basis to allow you to submit custom fields with CSR enrollments and PFX enrollments to supply custom request attributes to the CA during the enrollment process. This functionality offers such benefits as: • Preventing users from requesting invalid certificates, based on your specific certificate requirements per template. • Providing additional information to the CA with the CSR. Once created on the template, these values are shown in Keyfactor Command on the PFX and CSR enrollment pages in the *Additional Enrollment Fields* section. The fields are mandatory during enroll-ment. The data will appear on the CA / Issued Certificates attribute tab for certificates enrolled with a template configured with Keyfactor Command enrollment fields. **Note:** These are not metadata fields, so they are not stored in the Keyfactor Command database, but simply passed through to the CA. The CA in turn could, via a gateway or policy module, use this data to perform required actions. The array contains the following parameters:

Name	Description
Id	An integer indicating the ID of the custom enrollment field.
Name	A string indicating the name of the custom enrollment field. This name will appear on the enrollment pages.
Options	For multiple choice values, an array of strings containing the value choices.
DataType	An integer indicating the parameter type. The options are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>String: A free-form data entry field.</td></tr>
<tr><td>2</td><td>Multiple Choice: Provides a list of acceptable values for the field. The multiple choice values are provided in the Options parameter.</td></tr>
</table>
For example:
<pre>
"EnrollmentFields": [
   {
      "Id": 3,
      "Name": "MyCustomField",
      "Options": ["Green","Red","Yellow","Blue"],
      "DataType": 2
   }
]
</pre>
</td></tr>
<tr><td>MetadataFields</td><td>An array of objects containing template-level metadata field settings. Template-level metadata field configurations can override global metadata field configurations in these possible ways:
Configuration on the metadata field of required, optional or hidden.
The default value for the metadata field.
A regular expression defined for the field (string fields only) against which entered data will be validated along with its associated message.
For fields of data type multiple choice, the list of values that appear in multiple choice dropdowns.
Metadata field settings defined on a template apply to enrollments made with that template only. Template-level metadata field settings, if defined, take precedence over global-level metadata</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>field settings. The metadata field settings array contains the following parameters:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the template-specific metadata setting.</td></tr><tr><td>DefaultValue</td><td>A string containing the default value defined for the metadata field for the specific template.</td></tr><tr><td>MetadataId</td><td>An integer indicating the global metadata field associated with the template-specific settings.</td></tr><tr><td>Validation</td><td>A string containing the template-specific regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:<pre>^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$</pre>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”. This field is only supported for metadata</td></tr></table></td></tr></table>

Name	Description
	Name / **Description**
	fields with data type *string*.
Enrollment	An integer that indicates how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are: **Value** / **Description** 0 — Optional Users have the option to either enter a value or not enter a value in the field. 1 — Required Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page. 2 — Hidden The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.
Message	A string containing a message to present when a user enters information in a metadata field that does not match the template-specific regular expression (*Validation* field).

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example:
<pre>"MetadataFields": [
   {
      "Id": 4,
      "DefaultValue": "reggie.wallace@keyexample.com",
      "MetadataId": 4,
      "Validation": "^[a-zA-Z0-9'_\\.\\-]*@(keyexample\\.or-
g|keyexample\\.com)$",
      "Enrollment": 1,
      "Message": "Your email address must be of the form
user@keyexample.com or fname.lname@keyexample.com."
   },
   {
      "Id": 13,
      "DefaultValue": "E-Business",
      "MetadataId": 5,
      "Validation": "",
      "Enrollment": 0,
      "Message": "",
      "Options": "Accounting,E-Busi-
ness,Executive,HR,IT,Marketing,R&D,Sales"
   }
]</pre></td></tr>
<tr><td>Regexes</td><td>An array of objects containing the global template regular expression settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Regular expression details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>In integer indicating the Keyfactor Command reference ID of the certificate template the regular expression is associated with.</td></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against</td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**
	Name / **Description**
	which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated. Use the *GET /Templates/SubjectParts* method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts. The following are some regular expression examples:

Subject Part	Example

CN (Common Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Subject Part / **Example**
	one character in the Common Name field in the enrollment pages.
O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities:

Name	Description

Name	Description

Name	Description

Subject Part	Example
	`^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_`

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Subject Part / **Example**
	`\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$`

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Subject Part / **Example**
	This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods: `^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`
	IPv6 (Subject Alternative Name: IPv6 Address) — This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
	MAIL (Subject Alternative Name: Email) — This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>UPN (Subject Alternative Name: User Principal Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:

<code>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</code></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.

Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
</td></tr>
<tr><td>ExtendedKeyUsages</td><td>Currently not in use.</td></tr>
<tr><td>EnrollmentTemplatePolicy</td><td>An object containing the individual template-level template policy settings. Template policies defined on a template apply to enrollments made with that template only. Template-level policies, if defined, take precedence over system-wide template policies. For</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description** more information about system-wide template policies, see GET Templates Settings on page 2395. The template policy object contains the following parameters: **Value** / **Description** KeyInfo — An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate. **Name** / **Description** ECDSA — An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command. ECC curves may be specified using the well-known OIDs for ECC algorithms or by friendly name. Well-known OIDs include: • 1.2.840.10045.3.1.7 = P-

Name	Description
	Name / **Description** **Value** / **Description** **Name** / **Description** 256/prime256v1/secp256r1 • 1.3.132.0.34 = P-384/secp384r1 • 1.3.132.0.35 = P-521/secp521r1 When specifying by friendly name, do not include a slash (use “P-256”, not “P-256/prime256v1/secp256r1”). RSA: An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no curves for this type of key. Ed448: An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no

Name	Description
	Name / **Description** **Value** / **Description** **Name** / **Description** curves for this type of key. Ed25519 — An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no curves for this type of key. AllowKeyReuse — A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option applies to certificate renewals. By default, this is set to *true* at a system-wide level. AllowWildcards — A Boolean that indicates whether wildcards are allowed (true) or not (false). By default, this is set to *true* at a system-wide level. RFCEnforcement — A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to *true*, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to false at a system-wide level.</td></tr>
</table>
For example:
<pre>
"TemplatePolicy": {
   "AllowKeyReuse": false,
   "AllowWildcards": true,
   "RFCEnforcement": true,
   "KeyInfo": {
      "ECDSA": {
         "bit_lengths": [
            256,
            384,
            521
         ],
         "curves": [
            "1.2.840.10045.3.1.7",
            "1.3.132.0.34",
            "1.3.132.0.35"
         ]
      },
      "RSA": null,
      "Ed448": null,
      "Ed25519": null
   }
}
</pre>
</td></tr>
<tr><td>KeySize</td><td>A string indicating the minimum supported key size of the template.</td></tr>
<tr><td>Curve</td><td>A string indicating the OID of the elliptical curve algorithm configured for the template, for ECC templates.</td></tr>
</table>
</td></tr>
<tr><td>StandaloneCAs</td><td>An array of objects containing enrollment information for standalone certificate authorities available for enrollment for the current user. Information about the CA includes:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Name</td><td>The full name of the CA, made up of the DNS hostname of the certificate authority (e.g. myca.keyexample.com) and the logical name (e.g. CorpStandaloneCA1) for a full name similar to myca.keyexample.com\\CorpStandaloneCA1.</td></tr><tr><td>RFCEnforcement</td><td>A Boolean that sets whether certificate enrollments made through Keyfactor Command for this CA must include at least one DNS SAN (true) or not (false). In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. This setting at the CA level applies only to standalone CAs. For CAs that use templates, this setting is controlled at the template level and is ignored at the CA level.</td></tr><tr><td>SubscriberTerms</td><td>A Boolean that sets whether to add a checkbox on the enrollment pages to force users to agree to a custom set of terms before enrolling (true) or not (false). Tip: Configure a link to the custom terms using the URL to Subscriber Terms application setting. See Application Settings: Enrollment Tab on page 609 for more information.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.4 GET Enrollment Available Renewal ID

The GET /Enrollment/AvailableRenewal/ID/{id} method is used to check a specific certificate by ID to determine which renewal types are supported, if any. This method or the GET /Enrollment/AvailableRenewal/Thumbprint method can be used before using the POST /Enrollment/Renew method to make a determination as to which fields need to be submitted, depending on whether one-click renewal is supported. This method returns HTTP 200 OK on a success with the supported renewal type.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the note under the *CollectionId* input parameter, below.

Table 390: GET Enrollment Available Renewal ID {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer specifying the Keyfactor Command reference ID of the certificate on which to check the renewal status. Use the *GET /Certificates* method to determine the certificate ID. This information is also available in the certificate details for a certificate in the Keyfactor Command Management Portal.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Table 391: GET Enrollment Available Renewal ID {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>AvailableRenewalType</td><td>An integer indicating the supported renewal type. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>None—renewal is not supported for this certificate.</td></tr>
<tr><td>1</td><td>Seeded PFX is supported. A renewal can be done if a template and certificate authority are supplied in the renewal request, but one-click renewal is not supported.</td></tr>
<tr><td>2</td><td>One-click renewal is supported. A renewal can be done using the same template and certificate authority used in the original certificate, and a template and certificate authority do not need to be supplied in the renewal request.
One-click renewal is only supported if either one of the following is true:
The certificate is located together with its private key in one or more managed certificate store(s).
The certificate was enrolled with a template that has been configured in Keyfactor Command to allow private keys to be encrypted and stored in the Keyfactor Command database. See Certificate Templates on page 379 for more information.</td></tr>
</table>
Tip: If the AvailableRenewalType is 2, 1 is also supported for the certificate.</td></tr>
<tr><td>Message</td><td>A message providing more details about the available renewal type result (e.g. “One click renewal is not available for this certificate. Template does not have PFX enrollment enabled.”).</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.5 GET Enrollment Available Renewal Thumbprint

The GET /Enrollment/AvailableRenewal/Thumbprint/{thumbprint} method is used to check a specific certificate by thumbprint to determine which renewal types are supported, if any. This method or the GET /Enrollment/AvailableRenewal/ID method can be used before using the POST /Enrollment/Renew method to make a determination as to which fields need to be submitted, depending on whether one-click renewal is supported. This method returns HTTP 200 OK on a success with the supported renewal type.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions. See also the note under the *CollectionId* input parameter, below.

Table 392: GET Enrollment Available Renewal Thumbprint {thumbprint} Input Parameters

Name	In	Description
thumbprint	Path	**Required**. The thumbprint of the certificate on which to check the renewal status. Use the *GET /Certificates* method to determine the certificate thumbprint. This information is also available in the certificate details for a certificate in the Keyfactor Command Management Portal.
CollectionId	Query	An integer specifying an optional certificate collection identifier to validate that the user executing the request has sufficient permissions to do so. If a certificate collection ID is not supplied, the user must have global permissions to complete the action. Supplying a certificate collection ID allows for a check of the user's certificate collection-level permissions to determine whether the user has sufficient permissions at a collection level to complete the action. See Certificate Collection Permissions on page 627 for more information.

Table 393: GET Enrollment Available Renewal Thumbprint {thumbprint} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>AvailableRenewalType</td><td>An integer indicating the supported renewal type. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>None—renewal is not supported for this certificate.</td></tr>
<tr><td>1</td><td>Seeded PFX is supported. A renewal can be done if a template and certificate authority are supplied in the renewal request, but one-click renewal is not supported.</td></tr>
<tr><td>2</td><td>One-click renewal is supported. A renewal can be done using the same template and certificate authority used in the original certificate, and a template and certificate authority do not need to be supplied in the renewal request.
One-click renewal is only supported if either one of the following is true:
The certificate is located together with its private key in one or more managed certificate store(s).
The certificate was enrolled with a template that has been configured in Keyfactor Command to allow private keys to be encrypted and stored in the Keyfactor Command database. See <a>Certificate Templates on page 379 for more information.</td></tr>
</table>
Tip: If the AvailableRenewalType is 2, 1 is also supported for the certificate.</td></tr>
<tr><td>Message</td><td>A message providing more details about the available renewal type result (e.g. “One click renewal is not available for this certificate. Template does not have PFX enrollment enabled.”).</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.6 POST Enrollment CSR

The POST /Enrollment/CSR method is used to enroll for a certificate using a certificate signing request (CSR). This method returns HTTP 200 OK on a success with a message body containing a list of certificate details and any metadata that was associated with the certificate request.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/enrollment/csr/

Tip: Use the GET /Enrollment/CSR/Context/My method before this method to check which templates and CAs are available for enrollment for the requesting user before submitting the enrollment request.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

Table 394: POST Enrollment CSR Input Parameters

Name	In	Description
CSR	Body	**Required**. The base-64 encoded CSR that will be passed in for enrollment.
PrivateKey	Body	A string containing the base-64 encoded private key that corresponds to the CSR to be saved with the enrollment. This is done to support private key retention in Keyfactor Command for requests made through CSR enrollment. The key should be provided in unencrypted PKCS#8 format. The private key option is only supported for enrollments done using templates configured in Keyfactor Command for private key retention.
Timestamp	Body	**Required**. The current date and time. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Template	Body	**Required***. A string that sets the name of the certificate template that should be used to issue the certificate. The template short name should be used. This field is **required** unless the enrollment is being done against a standalone CA.
CertificateAuthority	Body	**Required***. A string that sets the name of the certificate authority that will be used to enroll against if there is more than one available with the provided template name. The certificate authority name can either be provided in *hostname**logical name* format or as just the *logical name*. For example: corpca01.keyexample.com\\CorpIssuingCA1 **OR** CorpIssuingCA1 If no certificate authority is provided, one will be chosen at random from the certificate authorities available for enrollment with the provided *Template*. This field is optional unless the enrollment is being done against a standalone CA, in which case it is **required**.
IncludeChain	Body	A Boolean that sets whether to include the certificate chain in the response (true) or not (false). The default is *false*.
Metadata	Body	An object of key/value pairs that set the values for the metadata fields that will be associated with the certificate

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>once it is in Keyfactor Command. The key is the field name and the value is the value for the field. For example:

<pre>"Metadata": {
   "AppOwnerFirstName": "William", // This is a String field.
   "AppOwnerLastName": "Smith",
   "AppOwnerEmailAddress": "william.smith@keyexample.com",
   "BusinessCritical": "true",   // This is a Boolean field.
   "BusinessUnit": "E-Business",   // This is a Multiple Choice field with a pre-defined value.
   "Notes": "Here are some notes.", // This is a BigText field.
   "SiteCode": 3,   // This is an integer field.
   "TicketResolutionDate": "2021-07-23"   // This is a Date field in yyyy-mm-dd format.
}</pre>
See <a>Certificate Metadata on page 710 for more information.</td></tr>
<tr><td>SANs</td><td>Body</td><td>An object that contains the elements for Keyfactor Command to use when generating the subject alternative name (SAN) for the certificate requested by the CSR, each of which is supplied as an array of strings. Possible values for the key are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>rfc822</td><td>RFC 822 Name</td></tr>
<tr><td>dns</td><td>DNS Name</td></tr>
<tr><td>directory</td><td>Directory Name</td></tr>
<tr><td>uri</td><td>Uniform Resource Identifier</td></tr>
<tr><td>ip4</td><td>IP v4 Address</td></tr>
<tr><td>ip6</td><td>IP v6 Address</td></tr>
<tr><td>registeredid</td><td>Registered ID (an OID)</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr>
<tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr>
</table>
For example:
<pre>"SANs": {
   "dns": [
      "dnssan1.keyexample.com",
      "dnssan2.keyexample.com",
      "dnssan3.keyexample.com"
   ],
   "ip4": [
      "192.168.2.73"
   ]
}</pre>
Note: Entering SANs with this option may either append or overwrite the SANs in the CSR request depending on how the issuing CA is configured. Please be sure to check that the certificate has the correct SANs after issuance. Any SAN added automatically as a result of the RFC 2818 compliance settings (see GET Templates on page 2422) will still be added alongside anything you add here. Review the SAN Attribute Policy Handler for the Keyfactor CA Policy Module (see Installing the Keyfactor CA Policy Module Handlers on page 2846) for more information.
</td></tr>
<tr><td>AdditionalEnrollmentFields</td><td>Body</td><td>An object that provide values for any custom enrollment fields set on the certificate template to supply custom request attributes to the CA during the enrollment process. For example:
<pre>"AdditionalEnrollmentFields": {
"CustomStringOne": "ValueOne",
"CustomMultiChoiceTwo": "ValueTwo" }</pre>
See Certificate Template Operations on page 381 for more</td></tr>
</table>

Name	In	Description
		information.
x-CertificateFormat	Header	**Required**. A string indicating the desired output format for the certificate. Available options are DER and PEM.

Table 395: POST Enrollment CSR Response Data

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateInformation</td><td>An object containing information about the certificate that was requested. CSR information includes:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SerialNumber</td><td>A string indicating the serial number of the certificate.</td></tr>
<tr><td>IssuerDN</td><td>A string indicating the issuer DN of the certificate.</td></tr>
<tr><td>Thumbprint</td><td>A string indicating the thumbprint of the certificate.</td></tr>
<tr><td>KeyfactorID</td><td>An integer indicating the Keyfactor Command reference ID of the issued certificate.</td></tr>
<tr><td>Certificates</td><td>An array of certificates in the order of:
end entity
intermediate CA
Root CA
Intermediate CA and root CA certificates will only be included if the request parameter IncludeChain was set to true.</td></tr>
<tr><td>WorkflowInstanceId</td><td>A string containing the Keyfactor Command reference GUID of the workflow instance.
Tip: Both the WorkflowInstanceId and the WorkflowReferenceId refer to the same workflow instance record—one using a GUID and one using a more human readable integer.</td></tr>
<tr><td>WorkflowReferenceId</td><td>An integer containing the Keyfactor Command reference ID of the workflow instance.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>Tip: Both the WorkflowInstanceId and the WorkflowReferenceId refer to the same workflow instance record—one using a GUID and one using a more human readable integer.</td></tr>
<tr><td>KeyfactorRequestId</td><td>An integer indicating the Keyfactor Command reference ID of the request.</td></tr>
<tr><td>RequestDisposition</td><td>A string indicating the state of the request (e.g. ISSUED).</td></tr>
<tr><td>DispositionMessage</td><td>A string providing a message regarding the enrollment (e.g. The private key was successfully retained.).</td></tr>
<tr><td>EnrollmentContext</td><td>An internally used Keyfactor Command field.</td></tr>
</table>
</td></tr>
<tr><td>Metadata</td><td>An array of the custom metadata values set on the certificate. The values vary depending on customization done in your environment. The information is presented in the following structure:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>MetadataFieldTypeName</td><td>A string containing the name of the metadata field in Keyfactor Command.</td></tr>
<tr><td>Value</td><td>The value of the metadata.</td></tr>
</table>
See Certificate Metadata on page 710 for more information.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.7 POST Enrollment PFX

The POST /Enrollment/PFX method is used to enroll for a certificate by supplying data in the desired fields. This method returns HTTP 200 OK on a success with a message body containing a list of certificate details and any metadata that was associated with the certificate request.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/enrollment/pfx/

Global or container-level schedule permissions for certificate stores are needed to install a certificate generated with this method into a certificate store (see the x-CertificateFormat on page 1679 parameter) using the POST /Enrollment/PFX/Deploy method (see POST Enrollment PFX Deploy on page 1685) or POST /Enrollment/PFX/Replace method (see POST Enrollment PFX Replace on page 1691).

Tip: Use the GET /Enrollment/PFX/Context/My method before this method to check which templates and CAs are available for enrollment for the requesting user before submitting the enrollment request.

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the POST /Enrollment/PFX method redesigns how enrollment flow works to handle require approval functionality in a Keyfactor Command workflow with support for delivery into certificate stores. Users who are planning to use require approval workflow functionality *and* deliver enrolled certificates into certificate stores must use version 2 of this endpoint.

Note: The supported key algorithms for a certificate template are determined based on global template policy, individual template policy, and the template's supported algorithm. When configuring template-level policies for key information, only key sizes that are valid for the algorithm will be available, according to the global template policy, the template policy, and the supported key sizes. For PFX and CSR enrollment, you must select a valid Key Length and Key Type for the enrollment.

Note: The *PopulateMissingValuesFromAD* parameter has been removed from the version 2 endpoint.

Table 396: POST Enrollment PFX v2 Input Parameters

Name	In	Description
AdditionalEnrollmentFields	Body	An object that provide values for any custom enrollment fields set on the certificate template to supply custom request attributes to the CA during the enrollment process. For example: `"AdditionalEnrollmentFields": { "CustomStringOne": "MyValue", "CustomMultiChoiceOne": "ValueTwo" }` See Certificate Template Operations on page 381 for more information.
CertificateAuthority	Body	**Required***. A string that sets the name of the certificate authority that will be used to enroll against if there is more than one available with the provided template name. The certificate authority name can either be provided in *hostname**logical name* format or as just the *logical name*. For example: corpca01.keyexample.com\\CorpIssuingCA1 **OR** CorpIssuingCA1 If no certificate authority is provided, one will be chosen at random from the certificate authorities available for enrollment with the provided *Template*. This field is optional unless the enrollment is being done against a standalone CA, in which case it is **required**.
ChainOrder	Body	A string indicating the order in which the certificate chain should be returned if *IncludeChain* is set to *true*. Supported values are *EndEntityFirst* or *RootFirst*.
Curve	Body	A string indicating the elliptic curve for the requested key. ECC curves may be specified using the well-known OIDs for ECC algorithms. Well-known OIDs include: • 1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1 • 1.3.132.0.34 = P-384/secp384r1 • 1.3.132.0.35 = P-521/secp521r1
CustomFriendlyName	Body	**Required***. A string that sets a custom friendly name for the certificate. This field is **required** if the *Require Custom Friendly Name* application setting is set to *true* (the default is *false*). See Application Settings: Enrollment Tab on page 609 for more information.
IncludeChain	Body	A Boolean that sets whether to include the certificate chain in

Name	In	Description
		the response (true) or not (false). The default is *true*.
InstallIn-toExistingCertificateStores	Body	A Boolean that sets whether to deploy the certificate to certificate stores (true) or not (false). The default is *true*. The *RenewalCertificateId* parameter is used in conjunction with *InstallIntoExistingCertificateStores* parameter to make the determination as to distribution of the certificate to certificate stores. If *InstallIntoExistingCertificateStores* is *true*, the certificate will be distributed to certificate stores that the certificate identified in *RenewalCertificateId* is found in.
KeyLength	Body	A string indicating the key size for the requested key. Supported key sizes are: • 255 • 256 • 384 • 448 • 521 • 2048 • 3072 • 4096 • 8192
KeyType	Body	A string indicating the algorithm for the request. Supported values are: • RSA • ECC • Ed448 • Ed25519

Metadata	Body	An object that set the values for the metadata fields that will be associated with the certificate once it is in Keyfactor Command. For example:
		`"Metadata": {`
		`"AppOwnerFirstName": "William", // This is a String field.`
		`"AppOwnerLastName": "Smith",`
		`"AppOwnerEmailAddress": "william.smith@keyexample.com",`
		`"BusinessCritical": "true", // This is a Boolean field.`

Name	In	Description

		<pre>"BusinessUnit": "E-Business", // This is a Multiple Choice field with a pre-defined value.
		"Notes": "Here are some notes.", // This is a BigText field.
		"SiteCode": 3, // This is an integer field.
		"TicketResolutionDate": "2021-07-23" // This is a Date field in yyyy-mm-dd format.
		}</pre>
		See Certificate Metadata on page 710 for more information.

Password	Body	**Required**. A string that sets the password used to encrypt the contents of the PFX file. The minimum password length is controlled by the *Password Length* application setting. The default is *12*. See Application Settings: Enrollment Tab on page 609 for more information.
RenewalCertificateId	Body	An integer that sets the ID of the certificate to be renewed when the method is called on a certificate renewal. The *RenewalCertificateId* parameter is used in conjunction with *InstallIntoExistingCertificateStores* parameter to make the determination as to distribution of the certificate to certificate stores. If *InstallIntoExistingCertificateStores* is *true*, the certificate will be distributed to certificate stores that the certificate identified in *RenewalCertificateId* is found in.
Stores	Body	An array of objects indicating the certificate stores to which the certificate should be distributed. Store details include: **Name** / **Description** StoreId — A string indicating the GUID of the certificate store to which the certificate should be deployed. Use the *GET /CertificateStores* method (see GET Certificate Stores on page 1327) with a query of "Approved -eq true" to retrieve a list of all your approved certificate stores to determine the GUID(s) of the store(s). Alias — A string containing the alias of the certificate upon entry into the store. The format of and

Name	In	Description

		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>requirement for this varies depending on the certificate store type and whether the *Overwrite* flag is selected. See PFX Enrollment on page 146 for more information.</td></tr><tr><td>Overwrite</td><td>A Boolean that sets whether a certificate in the store with the *Alias* provided should be overwritten with the new certificate (true) or not (false). The default is *false*.
		Use the *GET /Certificates/Locations/{id}* method (see GET Certificates Locations ID on page 1116) to retrieve a list of the locations an existing certificate is in to determine the alias used for the certificate in the certificate store.</td></tr><tr><td>Properties</td><td>An object indicating the unique parameters defined for the certificate store type that need to be populated for the certificate. The *key* is the name of the specific parameter from the certificate store type definition as returned in the JobProperties on the store type using the *GET CertificateStoreTypes* method and the *value* is the value that should be set for that parameter on the certificate in the certificate store. For example, for NetScaler, the key name that is optionally used to associate the certificate with a virtual server is *NetscalerVserver* and is returned by *GET CertificateStoreTypes* like so:
		<pre>"JobProperties": [
		"NetscalerVserver"]</pre>
		It can be seen in the Keyfactor Command Management Portal when editing the certificate store type in the field for *Management Job Custom Fields*.
		The setting is referenced using the following format:</td></tr></table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
"Properties":
{"NetscalerVserver":"MyVirtualSe
rverName"}
```

Note: The only built-in certificate store type that makes use of properties that can be set on a certificate-by-certificate basis in the store is NetScaler. You may have custom certificate store types that make use of this functionality.
</td></tr>
</table>
</td></tr>
<tr><td>Subject</td><td>Body</td><td>Required*. A string containing the subject name using X.500 format. For example:

```
"Subject":
"CN=websrvr14.keyexample.com,OU=IT,O=\"Key
Example, Inc.\",L=Independence,ST=OH,C=US"
```

This field is required if the Common Name Regular Expression application setting is set to the default value of .+. See <a>Application Settings: Enrollment Tab on page 609 for more information.</td></tr>
<tr><td>Timestamp</td><td>Body</td><td>Required. A string indicating the current date and time. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Template</td><td>Body</td><td>Required*. A string that sets the name of the certificate template that should be used to issue the certificate. The template short name should be used.
This field is required unless the enrollment is being done against a standalone CA.</td></tr>
<tr><td>SANs</td><td>Body</td><td>An object that contains the elements for Keyfactor Command to use when generating the subject alternative name (SAN) for the certificate requested by the CSR, each of which is supplied as an array of strings. Possible values for the key are:</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>rfc822</td><td>RFC 822 Name</td></tr>
<tr><td>dns</td><td>DNS Name</td></tr>
<tr><td>directory</td><td>Directory Name</td></tr>
<tr><td>uri</td><td>Uniform Resource Identifier</td></tr>
<tr><td>ip4</td><td>IP v4 Address</td></tr>
<tr><td>ip6</td><td>IP v6 Address</td></tr>
<tr><td>registeredid</td><td>Registered ID (an OID)</td></tr>
<tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr>
<tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr>
</table>
For example:
<pre>"SANs": {
   "dns": [
      "dnssan1.keyexample.com",
      "dnssan2.keyexample.com",
      "dnssan3.keyexample.com"
   ],
   "ip4": [
      "192.168.2.73"
   ]
}</pre>
</td></tr>
<tr><td>UseLegacyEncryption</td><td>Body</td><td>A Boolean that sets whether legacy encryption should be used in generating the PKCS#12 file resulting from the enrollment request (true) or not (false). Legacy encryption algorithms include PBE-SHA1-RC2-40 and PBE-SHA1-3DES. When legacy is not selected, encryption algorithms include AES-256-CBC, PBES2 with PBKDF2, and PBE-SHA1-3DES. The default is false.</td></tr>
<tr><td>x-CertificateFormat</td><td>Header</td><td>Required. A string containing the desired output format for the certificate. Available options are:
JKS
Select the JKS option when you intend to create a Java</td></tr>
</table>

Name	In	Description
		keystore with the returned PKCS12Blob. • PEM Select the PEM option when you intend to create a PEM file with the returned PKCS12Blob. • PFX Select the PFX option when you intend to create a PKCS#12 (PFX/P12) file with the returned PKCS12Blob. • Replace The Replace option is designed to be used when pushing an updated certificate to a certificate store (see POST Enrollment PFX Replace on page 1691). Selecting this item causes data to be staged in preparation for the replace step. • Store The Store option is designed to be used when pushing a newly obtained certificate to a certificate store (see POST Enrollment PFX Deploy on page 1685). Selecting this item causes data to be staged in preparation for the deploy step. • Zip Select the Zip option when you intend to output the returned PKCS12Blob as separate PEM-encoded certificate, private key, and optional chain certificate files wrapped together in a zip file.

Table 397: POST Enrollment PFX v2 Response Data

Value	Description
SuccessfulStores	An array of strings containing a comma delimited list of certificate stores, referenced by certificate store GUID, to which the certificate was successfully scheduled for deployment.
CertificateInformation	An object containing information about the certificate that was requested. Certificate information includes:

Value	Description
SerialNumber	A string indicating the serial number of the certificate.
IssuerDN	A string indicating the issuer DN of the certificate.
Thumbprint	A string indicating the thumbprint of the certificate.
KeyfactorID	An integer indicating the Keyfactor Command reference ID of the issued certificate.
PKCS12Blob	A string containing the base-64-encoded representation of the certificate in Zip or PFX format with the optional certificate chain. The string will need to be base-64 decoded for both Zip and PFX. This can be accomplished in a number of ways. For example, using PowerShell:

```
$b64 = Get-Content
'C:\path\to\source\file'
$targetFile =
'C:\path\to\target\file'

$bytes =
[Convert]::FromBase64String
($b64)
[IO.File]::WriteAllBytes
($targetFile, $bytes)
```

Value	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>**Note:** No value is returned for the PKCS12Blob if you select a certificate format of *Store* in the header. The *Store* option is designed to be used when pushing a newly obtained PFX certificate to a certificate store (see POST Enrollment PFX Deploy on page 1685).</td></tr><tr><td>Password</td><td>An internally used Keyfactor Command field.</td></tr><tr><td>WorkflowInstanceId</td><td>A string containing the Keyfactor Command reference GUID of the workflow instance. **Tip:** Both the *WorkflowInstanceId* and the *WorkflowReferenceId* refer to the same workflow instance record—one using a GUID and one using a more human readable integer.</td></tr><tr><td>WorkflowReferenceId</td><td>An integer containing the Keyfactor Command reference ID of the workflow instance. **Tip:** Both the *WorkflowInstanceId* and the *WorkflowReferenceId* refer to the same workflow instance record—one using a GUID and one using a more human readable integer.</td></tr><tr><td>KeyfactorRequestId</td><td>An integer indicating the Keyfactor Command reference ID of the request.</td></tr><tr><td>RequestDisposition</td><td>A string indicating the state of the request (e.g. ISSUED).</td></tr></table>

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>DispositionMessage</td><td>A string providing a message regarding the enrollment (e.g. The private key was successfully retained.).</td></tr>
<tr><td>EnrollmentContext</td><td>An internally used Keyfactor Command field.</td></tr>
</table>
</td></tr>
<tr><td>Metadata</td><td>An object containing the custom metadata values set on the certificate. The values vary depending on customization done in your environment. The information is presented in the following structure:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>MetadataFieldTypeName</td><td>A string containing the name of the metadata field in Keyfactor Command.</td></tr>
<tr><td>Value</td><td>The value of the metadata.</td></tr>
</table>
See <a>Certificate Metadata on page 710 for more information. </td></tr>
</table>

Version 1

Version 1 of the POST /Enrollment/PFX method includes the same capabilities as version 2 except when used in conjunction with Keyfactor Command workflows that require approval with an intended end goal of delivering the resulting certificate into a certificate store. In this specific case, version 2 must be used.

Table 398: POST Enrollment PFX v1 Input Parameters

Name	In	Description

AdditionalEnrollmentFields	Body	An object that provides values for any custom enrollment fields set on the certificate template to supply custom request attributes to the CA during the enrollment process. For example:
		`“AdditionalEnrollmentFields”: {`
		`“CustomStringOne”: "MyValue",`
		`“CustomMultiChoiceOne”: "ValueTwo" }`
		See Certificate Template Operations on page 381 for more information.

CertificateAuthority	Body	**Required***. A string that sets the name of the certificate authority that will be used to enroll against if there is more than one available with the provided template name. The certificate authority name can either be provided in *hostname**logical name* format or as just the *logical name*. For example:
		`corpca01.keyexample.com\\CorpIssuingCA1`
		`OR CorpIssuingCA1`
		If no certificate authority is provided, one will be chosen at random from the certificate authorities available for enrollment with the provided *Template*.
		This field is optional unless the enrollment is being done against a standalone CA, in which case it is **required**.

ChainOrder	Body	A string indicating the order in which the certificate chain should be returned if *IncludeChain* is set to *true*. Supported values are *EndEntityFirst* or *RootFirst*.
CustomFriendlyName	Body	**Required***. A string that sets a custom friendly name for the certificate. This field is **required** if the *Require Custom Friendly Name* application setting is set to *true* (the default is *false*). See Application Settings: Enrollment Tab on page 609 for more information.
Curve	Body	A string indicating the elliptic curve for the requested key. ECC curves may be specified using the well-known OIDs for ECC algorithms. Well-known OIDs include: • 1.2.840.10045.3.1.7 = P-

Name	In	Description
		256/prime256v1/secp256r1 • 1.3.132.0.34 = P-384/secp384r1 • 1.3.132.0.35 = P-521/secp521r1
IncludeChain	Body	A Boolean that sets whether to include the certificate chain in the response (true) or not (false). The default is *true*.
KeyLength	Body	A string indicating the key size for the requested key.
KeyType	Body	A string indicating the algorithm for the request. Supported values are: • RSA • ECC • Ed448 • Ed25519

Metadata	Body	An object that set the values for the metadata fields that will be associated with the certificate once it is in Keyfactor Command. The *key* is the field name and the *value* is the value for the field. For example:
		`"Metadata": {`
		`"AppOwnerFirstName": "William", // This is a String field.`
		`"AppOwnerLastName": "Smith",`
		`"AppOwnerEmailAddress": "williamsmith@keyexample.com",`
		`"BusinessCritical": "true", // This is a Boolean field.`
		`"BusinessUnit": "E-Business", // This is a Multiple Choice field with a pre-defined value.`
		`"Notes": "Here are some notes.", // This is a BigText field.`
		`"SiteCode": 3, // This is an integer field.`
		`"TicketResolutionDate": "2021-07-23" // This is a Date field in yyyy-mm-dd format.`
		`}`
		See Certificate Metadata on page 710 for more information.

Password	Body	**Required**. A string that sets the password used to encrypt the contents of the PFX file. The minimum password length is controlled by the *Password Length* applic-

Name	In	Description
		ation setting. The default is *12*. See Application Settings: Enrollment Tab on page 609 for more information.
PopulateMissingValuesFromAD	Body	A Boolean that sets whether to populate the information in the subject from Active Directory (true) or not (false). The default is *false*.
RenewalCertificateId	Body	An integer that sets the ID of the certificate to be renewed when the method is called on a certificate renewal.

SANs	Body	An object that contains the elements for Keyfactor Command to use when generating the subject alternative name (SAN) for the certificate requested by the CSR, each of which is supplied as an array of strings. Possible values for the key are:
		
		Value / **Description**
		rfc822 — RFC 822 Name
		dns — DNS Name
		directory — Directory Name
		uri — Uniform Resource Identifier
		ip4 — IP v4 Address
		ip6 — IP v6 Address
		registeredid — Registered ID (an OID)
		ms_ntprincipalname — MS_NTPrincipalName (a string)
		ms_ntdsreplication — MS_NTDSReplication (a GUID)
		
		For example:
		
		`"SANs": {`
		`  "dns": [`
		`  "dnssan1.keyexample.com",`

Name	In	Description

		<pre>"dnssan2.keyexample.com",
		"dnssan3.keyexample.com"
		],
		"ip4": [
		"192.168.2.73"
		]
		}</pre>

Subject	Body	**Required***. A string containing the subject name using X.500 format. For example:
		<pre>"Subject":
		"CN=websrvr14.keyexample.com,OU=IT,O=\"Key Example,
		Inc.\",L=Independence,ST=OH,C=US"</pre>
		This field is **required** if the *Common Name Regular Expression* application setting is set to the default value of *.+*. See Application Settings: Enrollment Tab on page 609 for more information.

Template	Body	**Required***. A string that sets the name of the certificate template that should be used to issue the certificate. The template short name should be used. This field is **required** unless the enrollment is being done against a standalone CA.
Timestamp	Body	**Required**. A string indicating the current date and time. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
UseLegacyEncryption	Body	A Boolean that sets whether legacy encryption should be used in generating the PKCS#12 file resulting from the enrollment request (true) or not (false). Legacy encryption algorithms include PBE-SHA1-RC2-40 and PBE-SHA1-3DES. When legacy is not selected, encryption algorithms include AES-256-CBC, PBES2 with PBKDF2, and PBE-SHA1-3DES. The default is *false*.
x-CertificateFormat	Header	**Required**. A string containing the desired output format for the certificate. Available options are: • JKS Select the JKS option when you intend to create a Java keystore with the returned PKCS12Blob.

Name	In	Description
		• PEM Select the PEM option when you intend to create a PEM file with the returned PKCS12Blob. • PFX Select the PFX option when you intend to create a PKCS#12 (PFX/P12) file with the returned PKCS12Blob. • Replace The Replace option is designed to be used when pushing an updated certificate to a certificate store (see POST Enrollment PFX Replace on page 1691). Selecting this item causes data to be staged in preparation for the replace step. • Store The Store option is designed to be used when pushing a newly obtained certificate to a certificate store (see POST Enrollment PFX Deploy on page 1685). Selecting this item causes data to be staged in preparation for the deploy step. • Zip Select the Zip option when you intend to output the returned PKCS12Blob as separate PEM-encoded certificate, private key, and optional chain certificate files wrapped together in a zip file.

Table 399: POST Enrollment PFX v1 Response Data

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateInformation</td><td>An object containing information about the certificate that was requested. Certificate information includes:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SerialNumber</td><td>A string indicating the serial number of the certificate.</td></tr>
<tr><td>IssuerDN</td><td>A string indicating the issuer DN of the certificate.</td></tr>
<tr><td>Thumbprint</td><td>A string indicating the thumbprint of the certificate.</td></tr>
<tr><td>KeyfactorID</td><td>An integer indicating the Keyfactor Command reference ID of the issued certificate.</td></tr>
<tr><td>PKCS12Blob</td><td>A string containing the base-64-encoded representation of the certificate in Zip or PFX format with the optional certificate chain. The string will need to be base-64 decoded for both Zip and PFX. This can be accomplished in a number of ways. For example, using PowerShell:
<pre>$b64 = Get-Content
'C:\path\to\source\file'
$targetFile =
'C:\path\to\target\file'

$bytes =
[Convert]::FromBase64String
($b64)
[IO.File]::WriteAllBytes
($targetFile, $bytes)</pre>
Note: No value is returned for the PKCS12Blob if you select a certificate format of Store in the header. The Store option is designed to be</td></tr>
</table>
</td></tr>
</table>

Value	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>used when pushing a newly obtained PFX certificate to a certificate store (see POST Enrollment PFX Deploy on page 1685).</td></tr><tr><td>Password</td><td>An internally used Keyfactor Command field.</td></tr><tr><td>WorkflowInstanceId</td><td>A string containing the Keyfactor Command reference GUID of the workflow instance. **Tip:** Both the *WorkflowInstanceId* and the *WorkflowReferenceId* refer to the same workflow instance record—one using a GUID and one using a more human readable integer.</td></tr><tr><td>WorkflowReferenceId</td><td>An integer containing the Keyfactor Command reference ID of the workflow instance. **Tip:** Both the *WorkflowInstanceId* and the *WorkflowReferenceId* refer to the same workflow instance record—one using a GUID and one using a more human readable integer.</td></tr><tr><td>KeyfactorRequestId</td><td>An integer indicating the Keyfactor Command reference ID of the request.</td></tr><tr><td>RequestDisposition</td><td>A string indicating the state of the request (e.g. ISSUED).</td></tr><tr><td>DispositionMessage</td><td>A string providing a message regarding the enrollment (e.g. The private key was successfully retained.).</td></tr><tr><td>EnrollmentContext</td><td>An internally used Keyfactor Command</td></tr></table>

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>field.</td></tr>
</table>
</td></tr>
<tr><td>Metadata</td><td>An object containing the custom metadata values set on the certificate. The values vary depending on customization done in your environment. The information is presented in the following structure:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>MetadataFieldTypeName</td><td>A string containing the name of the metadata field in Keyfactor Command.</td></tr>
<tr><td>Value</td><td>The value of the metadata.</td></tr>
</table>
See Certificate Metadata on page 710 for more information.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.8 POST Enrollment CSR Parse

The POST /Enrollment/CSR/Parse method takes a CSR in the body, parses it, and returns all elements that were found in the CSR. This method returns HTTP 200 OK on a success with the parsed CSR contents.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
None

Table 400: POST Enrollment CSR Parse Input Parameters

Name	In	Description
CSR	Body	**Required**. Base-64-encoded CSR with the Begin and End Certificate Request tags.

Table 401: POST Enrollment CSR Parse Response Data

Name	Description
(CSR Contents)	An array of strings in the form Name=Value containing all the elements in the CSR. Possible values include:

Name	Description
Key Length	An integer indicating the desired key size of the certificate.
Key Type	A string indicating the desired key encryption of the certificate.
CN	The common name of the certificate.
O	The organization of the certificate.
OU	The organizational unit of the certificate.
L	The city of the certificate.
ST	The state of the certificate.
C	The country (two characters) of the certificate.
E	The email address of the certificate.
DNS Name	A SAN value containing a DNS name.
IP Address	A SAN value containing an IP v4 or IP v6 address.
RFC822 Name	A SAN value containing an email message.
URL	A SAN value containing a uniform resource identifier.
Directory Name	A SAN value containing a directory name.
Registered ID	A SAN value containing a registered ID.
Other name:Principal Name	A SAN value containing a user principal name (UPN) value.
Other name:DS Object Guid	A SAN value containing the MS_NTDSReplication value.

Note: Some of these fields cannot be added to a CSR generated within Keyfactor

Name	Description
	Command (e.g. URL) and will only be found in CSRs generated outside Keyfactor Command.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.9 POST Enrollment PFX Deploy

The POST /Enrollment/PFX/Deploy method is used to put a certificate into a certificate store. It is intended to be used immediately after using the POST /Enrollment/PFX method to enroll for a PFX using the *Store* value for the *x-certificateformat* header (see POST Enrollment PFX on page 1665) or the POST /Enrollment/Renew method to renew a certificate already in a certificate store. This method returns HTTP 200 OK on a success with a message body containing the failed and succeeded stores.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/schedule/
/certificates/enrollment/pfx/
OR
/certificate_stores/schedule/#/ (where # is a reference to a specific certificate store container ID)
/certificates/enrollment/pfx/
Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Tip: The POST /Enrollment/PFX/Deploy method must be used within 5 minutes of acquiring a certificate with the POST /Enrollment/PFX or POST /Enrollment/Renew method as the same user who executed the certificate request. After 5 minutes, the temporary staging data needed in order to deploy the certificate is automatically cleared and is no longer available for deployment.

Table 402: POST Enrollment PFX Deploy Input Parameters

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>Stores</td><td>Body</td><td>Required*. An array of objects indicating the certificate stores to which the certificate should be deployed with additional properties as needed based on the store type and whether an existing certificate is being overwritten with the new certificate. Store parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreId</td><td>A string indicating the GUID of the certificate store(s) to which the certificate should be deployed.
Use the GET /CertificateStores method (see GET Certificate Stores on page 1327) with a query of “Approved -eq true” to retrieve a list of all your approved certificate stores to determine the GUID(s) of the store(s).</td></tr>
<tr><td>Alias</td><td>A string indicating the alias of the certificate upon entry into the store. The format of and requirement for this varies depending on the certificate store type and whether the Overwrite flag is selected. See PFX Enrollment on page 146 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Overwrite</td><td>A Boolean that sets whether a certificate in the store with the Alias provided should be overwritten with the new certificate (true) or not (false). The default is false.
Use the GET /Certificates/Locations/{id} method (see GET Certificates Locations ID on page 1116) to retrieve a list of the locations an existing certificate is in to determine the alias used for the certificate in the certificate store.</td></tr>
<tr><td>Properties</td><td>An object containing the unique parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for NetScaler, the key name that is optionally used to associate the certificate with a virtual server is NetscalerVserver and is returned by GET</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>CertificateStoreTypes like so:
<pre>"JobProperties": ["NetscalerVserver"]</pre>It can be seen in the Keyfactor Command Management Portal when editing the certificate store type in the field for Management Job Custom Fields.
The setting is referenced using the following format:
<pre>"Properties":
{"NetscalerVserver":"MyVirtualServerNam
e"}</pre>Note: The only built-in certificate store type that makes use of properties that can be set on a certificate-by-certificate basis in the store is NetScaler. You may have custom certificate store types that make use of this functionality.</td></tr>
</table>
This replaces the StoresIDs and StoreTypes parameters as of Keyfactor Command version 9.4.
</td></tr>
<tr><td>Password</td><td>Body</td><td>Required*. A string with a password used to secure the certificate in the certificate store. This field is required for store types that require an entry password, such as PEM stores.</td></tr>
<tr><td>CertificateId</td><td>Body</td><td>Required*. The integer for the certificate that needs to be deployed. This is returned in the response to the POST /Enrollment/PFX or POST /Enrollment/Renew request as the KeyfactorId.
Note: For enrollments that do not require manager approval (where the certificate is issued immediately), the CertificateId is required. The RequestId may be provided but is not required in this case. For enrollments that do require manager approval (where the certificate is not issued immediately), only the KeyfactorRequestId will be returned on the enrollment and the RequestId is required for deployment.</td></tr>
<tr><td>RequestId</td><td>Body</td><td>Required*. The integer of the request ID for the certificate that needs to be deployed. This is returned in the response to the POST /Enrollment/PFX or POST /Enrollment/Renew request as the KeyfactorRequestId.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>See the note under CertificateId regarding when this field is required and when it is not.</td></tr>
<tr><td>JobTime</td><td>Body</td><td>A string containing the date and time when the certificate should be deployed. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). Dates in the past will cause a management job to be created to run immediately. Dates in the future will result in a management job set to run in the future. The default is to create a management job that runs immediately.</td></tr>
<tr><td>StoreIds</td><td>Body</td><td>An array of strings containing the certificate store GUIDs for the stores to which the certificate should be added.
The StoreIds parameter is obsolete as of Keyfactor Command version 9.4 and has been replaced by the Stores parameter. It is still supported for backward compatibility, but no longer required.</td></tr>
<tr><td>StoreTypes</td><td>Body</td><td>An array of objects indicating the store types used with additional properties as needed based on the store type and whether an existing certificate is being overwritten with the new certificate.
The StoreTypes parameter is obsolete as of Keyfactor Command version 9.4 and has been replaced by the Stores parameter. It is still supported for backward compatibility, but is no longer required.
Store type parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreTypeId</td><td>An integer indicating the type of certificate store the certificate is being deployed to. The possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Java Keystore</td></tr>
<tr><td>2</td><td>PEM File</td></tr>
<tr><td>3</td><td>F5 SSL Profiles</td></tr>
<tr><td>4</td><td>IIS Roots</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>5</td><td>NetScaler</td></tr>
<tr><td>6</td><td>IIS Personal</td></tr>
<tr><td>7</td><td>F5 Web Server</td></tr>
<tr><td>8</td><td>IIS Revoked</td></tr>
<tr><td>9</td><td>F5 Web Server REST</td></tr>
<tr><td>10</td><td>F5 SSL Profiles REST</td></tr>
<tr><td>11</td><td>F5 CA Bundles REST</td></tr>
<tr><td>100</td><td>Amazon Web Services</td></tr>
<tr><td>101</td><td>File Transfer Protocol</td></tr>
<tr><td>1xx</td><td>User-defined certificate stores will be given a type ID over 101.</td></tr>
</table>
</td></tr>
<tr><td>Alias</td><td>A string indicating the alias of the certificate upon entry into the store. The format of and requirement for this varies depending on the certificate store type and whether the Overwrite flag is selected. See PFX Enrollment on page 146 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Overwrite</td><td>A Boolean that sets whether a certificate in the store with the Alias provided should be overwritten with the new certificate (true) or not (false). The default is false.
Use the GET /Certificates/Locations/{id} method (see GET Certificates Locations ID on page 1116) to retrieve a list of the locations an existing certificate is in to determine the alias used for the certificate in the certificate store.</td></tr>
<tr><td>Properties</td><td>An array of objects with the unique parameters</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description

		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>defined for the certificate store type that need to be populated for the certificate. The *key* is the name of the specific parameter from the certificate store type definition as returned in the JobProperties on the store type using the *GET CertificateStoreTypes* method and the *value* is the value that should be set for that parameter on the certificate in the certificate store. For example, for NetScaler, the key name that is optionally used to associate the certificate with a virtual server is *NetscalerVserver* and is returned by *GET CertificateStoreTypes* like so:
		<pre>"JobProperties": ["NetscalerVserver"]</pre>
		It can be seen in the Keyfactor Command Management Portal when editing the certificate store type in the field for *Management Job Custom Fields*.
		The setting is referenced using the following format:
		<pre>"Properties": [
		{"NetscalerVserver":"MyVirtualServerNam
		e"}]</pre>
		Note: The only built-in certificate store type that makes use of properties that can be set on a certificate-by-certificate basis in the store is NetScaler. You may have custom certificate store types that make use of this functionality.</td></tr></table>

Table 403: POST Enrollment PFX Deploy Response Data

Name	Description
SuccessfulStores	An array of strings containing the GUIDs for the certificates stores for which management jobs to deploy the certificate were successfully created. **Note:** Successful creation of a management job to deploy a certificate to a certificate store does not necessarily mean that a certificate will successfully be deployed to the store. A management job may fail for any number of reasons (e.g. permissions on the store). Use the *GET /Certificates/{id}* method with *includeLocations=true* to confirm that the certificate has successfully been deployed to the target store(s). The locations won't appear in the certificate record until after a certificate store inventory has been completed for each store.
FailedStores	An array of strings containing the GUIDs for the certificates stores for which management jobs to deploy the certificate could not be created.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.10 POST Enrollment PFX Replace

The POST /Enrollment/PFX/Replace method is used to replace a certificate in a certificate store. It is intended to be used immediately after using the POST /Enrollment/PFX method to enroll for a PFX using the *Replace* value for the *x-certificateformat* header (see POST Enrollment PFX on page 1665) or the POST /Enrollment/Renew method to renew a certificate already in a certificate store. This method returns HTTP 200 OK on a success with a message body containing the failed and succeeded stores.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_stores/schedule/
/certificates/enrollment/pfx/
OR
/certificate_stores/schedule/#/ (where # is a reference to a specific certificate store container ID)
/certificates/enrollment/pfx/

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Note: You could achieve the same end using the *POST /Enrollment/PFX/Deploy* method, but in that case you would need to provide the certificate store GUID(s), the alias of the current certificate in the certificate store(s), the certificate store type(s), and set the overwrite flag to true (as well as the certificate ID of the new certificate). To achieve a replacement with the *POST /Enrollment/PFX/Replace* method you only need to provide the certificate IDs of the certificate being replaced and the new certificate. All the rest of the work is done for you. The certificate will be replaced in all locations in which the certificate is found. If you want to replace the certificate in only some of the locations in which it is found, you will need to use the *POST /Enrollment/PFX/Deploy* method (see POST Enrollment PFX Deploy on page 1685).

Tip: The *POST /Enrollment/PFX/Replace* method must be used within 5 minutes of acquiring a certificate with the *POST /Enrollment/PFX* or POST /Enrollment/Renew method as the same user who executed the certificate request. After 5 minutes, the temporary staging data needed in order to deploy the certificate is automatically cleared and is no longer available for deployment.

Table 404: POST Enrollment PFX Replace Input Parameters

Name	In	Description
ExistingCertificateId	Body	**Required**. The integer of the certificate that will be replaced that is already in the store(s). A management job will be created to replace the certificate in all stores in which it is found. Use the *GET /Certificates* method to determine the certificate ID. This information is also available in the certificate details for a certificate in the Keyfactor Command Management Portal.
CertificateId	Body	**Required***. The integer for the certificate that needs to be deployed. This is returned in the response to the POST /Enrollment/PFX request. Either the *CertificateId* or the *RequestId* is **required** but not both.
RequestId	Body	**Required***. The integer of the request ID for the certificate that needs to be deployed. This is returned in the response to the POST /Enrollment/PFX request. Either the *CertificateId* or the *RequestId* is **required** but not both.
Password	Body	**Required***. A string with a password used to secure the certificate in the certificate store. This field is **required** for store types that require an entry password, such as PEM stores.
JobTime	Body	A string containing the date and time when the certificate should be deployed. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). Dates in the past will cause a management job to be created to run immediately. Dates in the future will result in a management job set to run in the future. The default is to create a management job that runs immediately.

Table 405: POST Enrollment PFX Replace Response Data

Name	Description
SuccessfulStores	An array of strings containing the GUIDs for the certificates stores for which management jobs to deploy the certificate were successfully created. **Note:** Successful creation of a management job to deploy a certificate to a certificate store does not necessarily mean that a certificate will successfully be deployed to the store. A management job may fail for any number of reasons (e.g. permissions on the store). Use the *GET /Certificates/{id}* method with *includeLocations=true* to confirm that the certificate has successfully been deployed to the target store(s). The locations won't appear in the certificate record until after a certificate store inventory has been completed for each store.
FailedStores	An array of strings containing the GUIDs for the certificates stores for which management jobs to deploy the certificate could not be created.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.16.11 POST Enrollment Renew

The POST /Enrollment/Renew method is used to enroll for a certificate renewal for a certificate that exists in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the new certificate. For certificates in a certificates store, this method does not automatically deploy the new certificate to the certificate store. In this case, the renew request should be followed by a call to either the POST /Enrollment/PFX/Deploy method or POST /Enrollment/PFX/Replace method to deploy the new certificate to the certificate store.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/collections/read/
/certificates/enrollment/pfx/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)
/certificates/enrollment/pfx/

Permissions for certificates can be set at either the global or certificate collection level. See Certificate Collection Permissions on page 627 for more information about global vs collection permissions.

Global or container-level schedule permissions for certificate stores are needed to install a certificate generated with this method into a certificate store using the POST /Enrollment/PFX/Deploy method (see POST Enrollment PFX Deploy on page 1685) or POST /Enrollment/PFX/Replace method (see POST Enrollment PFX Replace on page 1691).

Note: As of Keyfactor Command version 10, enrollment (PFX and CSR), renewal, and revocation requests all flow through Keyfactor Command workflow. This will result in no changes to the enrollment, renewal, and revocation user experience unless customizations have been added in workflow (see Workflow Definitions on page 230).

Table 406: POST Enrollment Renew Input Parameters

Name	In	Description
CertificateId	Body	**Required***. The integer for the certificate in Keyfactor Command that needs to be renewed. Either the *CertificateId* or the *Thumbprint* is **required** but not both.
Thumbprint	Body	**Required***. The thumbprint for the certificate that needs to be renewed. Either the *CertificateId* or the *Thumbprint* is **required** but not both.
Timestamp	Body	**Required**. The current date and time. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
CertificateAuthority	Body	**Required***. A string that sets the name of the certificate authority that will be used to enroll against. The certificate authority name should be provided in *hostname**logical name* format. For example: `corpca01.keyexample.com\\CorpIssuingCA1` This field is **required** if one-click renewal is not supported for the certificate (see GET Enrollment Available Renewal ID on page 1653 or GET Enrollment Available Renewal Thumbprint on page 1656).
Template	Body	**Required***. A string that sets the name of the certificate template that should be used to issue the certificate. The template short name should be used. This field is **required** if one-click renewal is not supported for the certificate (see GET Enrollment Available Renewal ID on page 1653 or GET Enrollment Available Renewal Thumbprint on page 1656).

Table 407: POST Enrollment Renew Response Data

Name	Description
KeyfactorID	ID of the certificate in Keyfactor Command.
KeyfactorRequestID	ID of the request in Keyfactor Command.
Thumbprint	Thumbprint of the certificate.
SerialNumber	Serial number of the certificate.
IssuerDN	Issuer DN of the certificate.
RequestDisposition	State of the request (e.g. issued).
DispositionMessage	Enrollment message (e.g. The private key was successfully retained.).
Password	A password generated for convenience for use on installation to a certificate store. This password may be used when deploying the certificate to a certificate store using the POST /Enrollment/Deploy method, though an alternate password may be used. The passwords do not need to match.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.17 Event Handler Registration

EventHandlerRegistration endpoints includes methods necessary to list, add, edit and delete event handler registrations for Alerts in Keyfactor Command.

Table 408: EventHandlerRegistration Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes an event handler.	DELETE Event Handler Registration ID on page 1704
/{id}	GET	Returns a registered event handler that matches the provided ID.	GET Event Handler Registration ID on page 1701
/{id}	PUT	Updates a registered event handler's information.	PUT Event Handler Registration ID on page 1702
/	GET	Returns all registered event handlers according to the provided filter and output parameters.	GET Event Handler Registration below
/	POST	Registers an event handler.	POST Event Handler Registration on page 1700

3.6.17.1 GET Event Handler Registration

The GET /EventHandlerRegistration method is used to retrieve a list of the Event Handler Registrations in Keyfactor Command (see Event Handler Registration on page 739). This method returns HTTP 200 OK on a success with details of the event handler. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/handlers/registration/read/

Table 409: GET Event Handler Registration Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • *DisplayName* • *Enabled*
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 410: GET Event Handler Registration Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the event handler.
DisplayName	A string indicating the display name of the event handler.
ClassName	A string indicating the class name of the event handler (for example, *CSS.CMS.Monitoring.EventHandling.Denied.DeniedLogger*).
Enabled	A Boolean indicating whether the event handler is enabled (true) or not (false).
SupportedEvents	A string indicating which application events the event handler supports. Built-in events include: • Certificate Expiration Handler • Denied Certificate Request Handler • Issued Certificate Handler • Key Rotation Handler • Pending Certificate Handler

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.17.2 POST Event Handler Registration

The POST /Event Handler Registration method is used to register an event handler (see Event Handler Registration on page 739). This method returns HTTP 200 OK on a success with details of the added event handler(s).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/handlers/registration/modify/

Table 411: POST Event Handler Registration Input Parameters

Name	In	Description
AssemblyName	Body	**Required**. A string containing the assembly name of the event handler to register. The handler file must be in the configured extensions directory on the machine running the Management Portal (see Application Settings: Console Tab on page 602).

Table 412: POST Event Handler Registration Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the event handler.
DisplayName	A string indicating the display name of the event handler.
ClassName	A string indicating the class name of the event handler (for example, *CSS.CMS.Monitoring.EventHandling.Denied.DeniedLogger*).
Enabled	A Boolean indicating whether the event handler is enabled (true) or not (false).
SupportedEvents	A string indicating which application events the event handler supports. Built-in events include: • Certificate Expiration Handler • Denied Certificate Request Handler • Issued Certificate Handler • Key Rotation Handler • Pending Certificate Handler

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.17.3 GET Event Handler Registration ID

The GET /EventHandlerRegistration/{id} method is used to retrieve a registered event handler that matches the provided ID in Keyfactor Command (see Event Handler Registration on page 739). This method returns HTTP 200 OK on a success with details of the specified event handler.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/handlers/registration/read/

Table 413: GET Event Handler Registration {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the event handler. Use the *GET /EventHandlerRegistration* method (see GET Event Handler Registration on page 1698) to retrieve a list of all the event handlers to determine the ID.

Table 414: GET Event Handler Registration Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the event handler.
DisplayName	A string indicating the display name of the event handler.
ClassName	A string indicating the class name of the event handler (for example, *CSS.CMS.Monitoring.EventHandling.Denied.DeniedLogger*).
Enabled	A Boolean indicating whether the event handler is enabled (true) or not (false).
SupportedEvents	A string indicating which application events the event handler supports. Built-in events include: • Certificate Expiration Handler • Denied Certificate Request Handler • Issued Certificate Handler • Key Rotation Handler • Pending Certificate Handler

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.17.4 PUT Event Handler Registration ID

The PUT /EventHandlerRegistration/{id} method is used to update the indicated registered event handler's *DisplayName* or *Enabled* status (see Event Handler Registration on page 739). This method returns HTTP 200 OK on a success with details of the updated event handler.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/handlers/registration/modify/

Table 415: PUT Event Handler Registration {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the event handler. Use the *GET /EventHandlerRegistration* method (see GET Event Handler Registration on page 1698) to retrieve a list of all the event handlers to determine the ID.
DisplayName	Body	A string indicating the display name of the event handler.
Enabled	Body	A Boolean indicating whether the event handler is enabled (true) or not (false).

Table 416: PUT Event Handler Registration {id} Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the event handler.
DisplayName	A string indicating the display name of the event handler.
ClassName	A string indicating the class name of the event handler (for example, *CSS.CMS.Monitoring.EventHandling.Denied.DeniedLogger*).
Enabled	A Boolean indicating whether the event handler is enabled (true) or not (false).
SupportedEvents	A string indicating which application events the event handler supports. Built-in events include: • Certificate Expiration Handler • Denied Certificate Request Handler • Issued Certificate Handler • Key Rotation Handler • Pending Certificate Handler

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.17.5 DELETE Event Handler Registration ID

The DELETE /EventHandlerRegistration/{id} method is used to delete the event handler with the provided ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/handlers/registration/modify/

Table 417: DELETE Event Handler Registration {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the event handler. Use the *GET /EventHandlerRegistration* method (see GET Event Handler Registration on page 1698) to retrieve a list of all the event handlers to determine the ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.18 Extensions Scripts

The Extensions Scripts component of the Keyfactor API includes methods necessary to create, update, retrieve and delete scripts configured for certificate requests, SSH alerts, and workflows. The Extensions Scripts endpoints are new as of Keyfactor Command version 11 and replace previous functionality which stored scripts for alert event handlers and workflow PowerShell steps in files in the directory structure on the server. The new method of storing the scripts in the database ensures they are stored safely and accessed only with appropriate permissions. Also see PowerShell Scripts on page 219 for important information about working with scripts in Keyfactor Command.

Important: When upgrading from a version of Keyfactor Command prior to version 11, the upgrade process will search the file location defined in the *Application settings > Console Tab > Extension Handler Path* setting and add all the files found in that directory to the database with the naming convention of *foldername (_subfolder name, if applicable)_filename* so it is clear which scripts were imported from which location (e.g., net6.0_Workflow_CustomPowerShellExample). The upgrade process will also identify which, if any, of the categories the script is configured for and add that information to the database with the script.

Table 418: Extensions Scripts Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes a script.	DELETE Extensions Scripts ID below
/{id}	GET	Returns a single script that matches the provided ID.	GET Extensions Scripts ID on the next page
/	GET	Returns all scripts according to the provided filter and output parameters.	GET Extensions Scripts on page 1707
/	POST	Adds a new script.	POST Extensions Scripts on page 1709
/	PUT	Updates a script.	PUT Extensions Scripts on page 1713

3.6.18.1 DELETE Extensions Scripts ID

The DELETE /Extensions/Scripts/{id} method is used to delete a script. Scripts cannot be deleted if configured for an alert or workflow. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/scripts/modify/

Table 419: DELETE Extensions Scripts Input Parameters

Name	In	Description
id	Path	**Required**. In integer indicating the Keyfactor Command reference ID for the script to be deleted. Use the *GET /Extensions/Scripts* method (see GET Extensions Scripts on page 1707) to retrieve a list of all the scripts to determine the ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.18.2 GET Extensions Scripts ID

The GET /Extensions/Scripts/{id} method is used to return the details of the script that matches the provided ID. This method returns HTTP 200 OK on a success with details of the specified script record.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/scripts/read/

Table 420: GET Extensions Scripts {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the script to be retrieved. Use the *GET /Extensions/Scripts* method (see GET Extensions Scripts on the next page) to retrieve a list of all the scripts to determine the ID.

Table 421: GET Extensions Scripts {id} Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the script.
Name	A string indicating the user-defined name of the script. The name of a script cannot be changed once posted.
Contents	A JSON-escaped string containing the contents of the script on a single line.
Categories	An array of either integers or case-sensitive strings indicating which category or categories the script applies to. The category of a script cannot be changed if it is in use in any alerts or workflows of that category. Possible category values are:

Code	Category Name
1	Expiration
2	Pending
3	Denied
4	Issued
5	KeyRotation
6	Workflow

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.18.3 GET Extensions Scripts

The GET /Extensions/Scripts method is used to return a list of all scripts configured in Keyfactor Command according to the provided filter and output parameters. This method returns HTTP 200 OK on a success with details of the script records.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/scripts/read/

Table 422: GET Extensions Scripts Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • *Name* • *Category* Use the equal (-eq) operator; the -ne operator is not supported. For example: *Category -eq "Expiration"* will return any scripts that have Expiration in their categories.
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 423: GET Extensions Scripts Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the script.
Name	A string indicating the user-defined name of the script. The name of a script cannot be changed once posted.
Categories	An array of either integers or case-sensitive strings indicating which category or categories the script applies to. The category of a script cannot be changed if it is in use in any alerts or workflows of that category. Possible category values are:

Code	Category Name
1	Expiration
2	Pending
3	Denied
4	Issued
5	KeyRotation
6	Workflow

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.18.4 POST Extensions Scripts

The POST /Extensions/Scripts method is used to add a new script to the database. This method returns HTTP 200 OK on a success with details of the newly created script record.

Important: This is the only means to add a script to the database. There is no UI equivalent for security reasons. (Upgrading from a version previous to version 11 will import existing scripts into the database as the only other means of adding scripts to the database).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/scripts/modify/

Table 424: POST Extensions Scripts Input Parameters

Name	In	Description
Name	Body	**Required**. A string indicating the user-defined name of the script. The name of a script cannot be changed once posted.
Contents	Body	**Required**. A JSON-escaped string containing the contents of the script on a single line. **Tip:** See below for examples of creating and handling this string.
Categories	Body	**Required**. An array of either integers or case-sensitive strings indicating which category or categories the script applies to. The category of a script cannot be changed if it is in use in any alerts or workflows of that category. Possible category values are:

Code	Category Name
1	Expiration
2	Pending
3	Denied
4	Issued
5	KeyRotation
6	Workflow

Two Approaches to Importing Scripts to the Database

To create a string value for the *Contents* field, you need to take your script, turn it into a string, and JSON-escape the string so that CR/LFs, tabs and the like will be encoded appropriately and the string will be on a single line. For example, the following string contains escaped CR/LFs (\r\n):

```
$MyVar = \"Hello, World!\"\r\n\r\nWrite-Host $MyVar
```

One approach to doing this uses a PowerShell script similar to the following, which takes the script to be uploaded as input and creates an output file with the JSON-escaped string:

```
# Path to the input file
$filePath = "C:\Stuff\UpdateSubjectSANs.ps1"

# Get the contents of the input file as a string (as opposed to an array of strings) into a variable
$fileContent = Get-Content -Path $filePath -Raw

# Set variables for the other body parameters - for multiple categories, use commas (e.g. 2,3,4)
$ScriptName = "UpdateSubjectSANs"
[int32[]]$Categories = 6

# Build the body
$body = @{
   "Name" = $ScriptName
   "Contents" = $fileContent.ToString()
   "Categories" = $Categories
}

# JSON escape the body elements
$JSONbody = ConvertTo-JSON $body

# Output the body elements including the escaped Contents string to a file
Set-Content -Value $JSONbody -Path C:\Stuff\MyOutFile.txt
```

The contents of the output file will look something like (the Contents field is shown truncated here):

```
{
   "Categories":  [
      6
   ],
   "Contents":  "# Declare your parameters at the beginning ($CSRSubject, $CSRSANs)\r\nparam(\r\n
[string]$CSRSubject,\r\n   [string]$CSRSANs,\r\n",
   "Name":  "UpdateSubjectSANs"
}
```

You can then open the output file, display the content without line wrapping the Contents field, and copy either the entire body or the JSON-escaped Contents string for pasting into your API command. Any line wraps that display on the screen in the Contents field will be interpreted by copy/paste as CR/LF, which will cause the API command to fail. If your script is long, you will need to be sure to use a text editor to open the file that can display the entire length of the Contents string as a single line. The built-in Windows Notepad application will display a maximum of 1024 characters on a line before wrapping even if word wrap is disabled. A tool such as the third-party Notepad++ is much less limited.

Alternately, you can do the JSON-escaping and update to the Keyfactor Command database in a single PowerShell script and skip the file output with copy/paste. The following script will JSON-escape a script and add it to the database:

```
# Prompt for credentials to authenticate to the Keyfactor API
$cred = Get-Credential

# Encode credentials (assumes the Keyfactor API is using Basic authentication)
$pair = "$($cred.Username):$($cred.GetNetworkCredential().Password)"
$encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
$basicAuthValue = "Basic $encodedCreds"

# Path to the input file
$filePath = "C:\Stuff\UpdateSubjectSANs.ps1"

# Keyfactor Command server name and optional port for API request
$APIServer = "keyfactor.keyexample.com"

# Set variables for the other body parameters - for multiple categories, use commas (e.g. 2,3,4)
$ScriptName = "UpdateSubjectSANs"
[int32[]]$Categories = 6

# Get the contents of the input file as a string (as opposed to an array of strings) into a variable
$fileContent = Get-Content -Path $filePath -Raw

# Build the headers
$headers = @{
   "Authorization"=$basicAuthValue
   "Accept"="application/json"
   "x-keyfactor-requested-with"="APIClient"
}

# Build the body
$body = @{
   "Name" = $ScriptName
   "Contents" = $fileContent.ToString()
   "Categories" = $Categories
}

# Make the API request to create a new script in the database
Invoke-WebRequest -Uri "https://$APIServer/KeyfactorAPI/Extensions/Scripts" -Method:Post -Headers
$headers -ContentType "application/json" -Body ($body|ConvertTo-Json) -ErrorAction:Stop -TimeoutSec
60
```

Table 425: POST Extensions Scripts Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the script.
Name	A string indicating the user-defined name of the script. The name of a script cannot be changed once posted.
Contents	A JSON-escaped string containing the contents of the script on a single line.
Categories	An array of either integers or case-sensitive strings indicating which category or categories the script applies to. The category of a script cannot be changed if it is in use in any alerts or workflows of that category. Possible category values are:

Code	Category Name
1	Expiration
2	Pending
3	Denied
4	Issued
5	KeyRotation
6	Workflow

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.18.5 PUT Extensions Scripts

The PUT /Extensions/Scripts method is used to update a script. This method returns HTTP 200 OK on a success with details of the updated script record.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/scripts/modify/

Note: You cannot change the name of a script once it has been created, so that field is not available as an input parameter to the PUT method.

Table 426: PUT Extensions Scripts Input Parameters

Name	In	Description
id	Body	**Required**. An integer indicating the Keyfactor Command reference ID for the script. Use the *GET /Extensions/Scripts* method (see GET Extensions Scripts on page 1707) to retrieve a list of all the scripts to determine the ID.
Contents	Body	A JSON-escaped string containing the contents of the script on a single line. If the contents field is not provided or is an empty string, the field will be ignored. (The contents of a script in the database cannot be cleared.) **Tip:** See POST Extensions Scripts on page 1709 for examples of creating and handling this string.
Categories	Body	**Required**. An array of either integers or case-sensitive strings indicating which category or categories the script applies to. The category of a script cannot be changed if it is in use in any alerts or workflows of that category. Possible category values are: **Code** / **Category Name** 1 — Expiration 2 — Pending 3 — Denied 4 — Issued 5 — KeyRotation 6 — Workflow **Tip:** The list of categories provided will completely replace any previously supported categories for the script. However, you cannot remove a category if the script is configured to be used by that category. You can add additional categories to a script that is already in use by select categories by including the existing categories in the parameter entry and adding any others as desired.

Table 427: PUT Extensions Scripts Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID for the script.
Name	A string indicating the user-defined name of the script. The name of a script cannot be changed once posted.
Contents	A JSON-escaped string containing the contents of the script on a single line.
Categories	An array of either integers or case-sensitive strings indicating which category or categories the script applies to. The category of a script cannot be changed if it is in use in any alerts or workflows of that category. Possible category values are:

Code	Category Name
1	Expiration
2	Pending
3	Denied
4	Issued
5	KeyRotation
6	Workflow

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.19 Identity Providers

The Identity Providers component of the Keyfactor API includes methods necessary to programmatically edit and retrieve identity providers within Keyfactor Command.

Table 428: Identity Providers Endpoint

Endpoint	Method	Description	Link
/{id}	GET	Returns the identity provider with the specified ID.	GET Identity Providers ID on the

Endpoint	Method	Description	Link
			next page
/{id}	PUT	Updates the identity provider with the specified ID.	PUT Identity Providers ID on page 1730
/	GET	Returns all identity providers defined within Keyfactor Command with filtering and output options.	GET Identity Providers on page 1760
/Types	GET	Returns details for all the identities provider types defined within Keyfactor Command.	GET Identity Providers Types on page 1775

3.6.19.1 GET Identity Providers ID

The GET /Identity/Providers/{id} method is used to return an identity provider by ID. This method returns HTTP 200 OK on a success with details for the specified identity provider.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/identity_providers/read/

Table 429: GET Identity Providers{id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the identity provider to retrieve. Use the *GET /Identity/Providers* method (see GET Identity Providers on page 1760) to retrieve a list of all the identity providers to determine the provider's ID.

Table 430: GET Identity Providers {id} Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the identity provider.
Authentic-ationScheme	A string indicating the authentication scheme for the identity provider.
DisplayName	A string indicating the display name for the identity provider.
TypeId	A string indicating the Keyfactor Command reference GUID for the type of identity provider. Possible values are: • DFB94650-E4EB-402A-B807-4F3CC91F712D (Active Directory) • F96B6464-11B7-4499-BEA7-B5AA6BA1571D (Generic—select this for Keyfactor Identity Provider) • 5AA04122-CD7C-48BA-AC11-F39E30AE8720 (Auth0)
Parameters	An array of objects containing information for each parameter set for the identity provider. Each parameter (Table 431: Identity Provider Parameters) contains the data shown in Table 432: Identity Provider Parameter Structure. Each parameter (Table 431: Identity Provider Parameters) contains the data shown in Table 432: Identity Provider Parameter Structure.

Table 431: Identity Provider Parameters

Name	Type	Example	Description
Admin Querying Client Id	1 - String	Command-API-Query	The client ID of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). Keyfactor recommends that you use a different client for this purpose than the client used for the main connection from Keyfactor Command to the identity provider (see *Client Id*). This parameter is required.

Name	Description

Name	Type	Example	Description
Admin Querying Client Secret	1 - String		The client secret of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). This parameter is required.
OIDC Audience	1 - String	Command-OIDC-Client	The audience value for the identity provider. For Keyfactor Identity Provider, this should be set to the same value as the *Client Id*. For example: `Command-OIDC-Client` This parameter is required.
Auth0 API URL	1 - String		The unique identifier defined in Auth0 or a similar identity provider for the API. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Authority	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor	The issuer/authority endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and

Name	Description

Name	Type	Example	Description
			Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required. **Tip:** When you add or update an identity provider, the provider's discovery document is validated based on this authority URL. The discovery document is also validated periodically in the background. The following are validated: • That the discovery document is reachable using the Authority value provided and can be parsed into a valid discovery document. • That the Authority URL matches the Issuer returned in the discovery document. • That all the URLs on the discovery document are using HTTPS. • That the JSONWebKeySetUri value is

Name	Description

Name	Type	Example	Description
			included on the discovery document. • That any endpoint configuration values (Authorization Endpoint, Token Endpoint, UserInfo Endpoint, JSONWebKeySetUri) that have been saved or are being saved match—including case—the values returned in the discovery document. The UserInfo Endpoint is not a required configuration field, but if a value is provided, it must match what's in the discovery document. If any of these validation tests fail, any identity provider changes in process will not be saved and an error will be displayed or logged.
Authorization Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/auth	The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can

Name	Description

Name	Type	Example	Description
			be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Client Id	1 - String	Command-OIDC-Client	The ID of the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, this should be: `Command-OIDC-Client` For more information, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). This parameter is required.
Client Secret	2 - Secret		The secret for the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716 for help locating this. It is automatically returned by the *Discovery Document*

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
Endpoint Fetch step. Review it to confirm that it appears correct.
Supported methods to store secret information are:
• Store the secret information in the Keyfactor secrets table.
A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.
• Load the secret information from a PAM provider.
See <a>Privileged Access Management (PAM) on page 742 for more information.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the secret. This parameter is used when PAM is not used as the storage location.</td></tr>
<tr><td>Parameters</td><td>An object</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>indicating the parameters to supply for PAM authentication. These will vary depending on the PAM provider.</td></tr>
<tr><td>Provider</td><td>A string indicating the ID of the PAM provider.
Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the ID.</td></tr>
</table>
For example, a username stored as a Keyfactor secret will look like:
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description

```
{
   "SecretValue":
"KEYEXAMPLE\svc_MySer-
viceName"
}
```

For example, a password stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "MySu-
perSecretPassword"
}
```

A secret stored as a CyberArk PAM secret will look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898 and the Safe, Folder, and Object reference the information in the CyberArk safe needed for this record):

```
{
   "Provider": "1",
   "Parameters":{
      "Safe":"MySafeName",
      "Folder":"MyFolder-
Name",
      "Object":"MyOb-
jectName"
   }
}
```

A secret stored as a Delinea PAM secret will look like (where the Provider value—2 in this

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>example—is the Id value from GET PAM Providers on page 1898 and the SecretId and SecretFieldName contain the information created in the Delinea secret server for this purpose):

<pre>{
   "Provider": "2",
   "Parameters":{
      "SecretId":"MyId"

"SecretFieldName":"MyReferenceName"
   }
}</pre>
This parameter is required.</td></tr>
<tr><td>Discovery Document Endpoint</td><td>1 - String</td><td>https://my-keyidp-server.keyexample.com/realms/Keyfactor/.well-known/openid-configuration</td><td>The discovery URL for the identity provider.
For Keyfactor Identity Provider, this is the link to the OpenID Endpoint Configuration page, which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). Populate this value and click Fetch to populate the remainder of the fields in this section, if desired.
If you opt not to populate this field or if the discovery document does not return a valid response, the remainder of the fields in this section of the configuration will need to be</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description
			configured manually. This value is not stored in the database.
Fallback Unique Claim Type	1 - String	cid	A backup value used to reference the type of claim used for users in the identity provider in case the primary referenced name does not contain a value. This parameter is required.
JSON Web Key Set Uri	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/certs	The JWKS (JSON Web Key Set) URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Name Claim Type	1 - String	preferred_username	The name used to reference the type of user claim for the identity provider. For Keyfactor Identity Provider, this should be: `preferred_username` This parameter is required.
Role	1 -	groups	The value used to reference the

Name	Description

Name	Type	Example	Description
Claim Type	String		type of group claim for the identity provider. For Keyfactor Identity Provider, this should be: `groups` This parameter is required.
OIDC Scope	1 - String		One or more scopes that are requested during the OIDC protocol when Keyfactor Command is the relying party. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Sign Out URL	1 - String	https://my-auth0-instance.us.auth0.com/oidc/logout	The signout URL for the identity provider. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Timeout	1 - String	60	The number of seconds a request to the identity provider is allowed to process before timing out with an error.
Token Audience	1 - String		An audience value to be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. This value is not used for Keyfactor Identity Provider.
Token	1 -	https://my-keyidp-serv-	The token endpoint URL for the

Name	Description

Name	Type	Example	Description
Endpoint	String	er.keyexample.com/realms/Keyfactor/protocol/openid-connect/token	identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Token Scope	1 - String		One or more scopes that should be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Unique Claim Type	1 - String	sub	The value used to reference the type of claim used for users in the identity provider. For Keyfactor Identity Provider, this should be (for subject): `sub` This parameter is required.
User Info	1 -	https://my-keyidp-serv-	The user info endpoint URL for

Name	Description

Name	Type	Example	Description
Endpoint	String	er.keyexample.com /realms/Keyfactor/protocol/openid-connect/certs	the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct.
User Query Endpoint	1 - String	https://my-keyidp-server.keyexample.com /admin/realms/Keyfactor	The user query endpoint URL for the identity provider. For Keyfactor Identity Provider, this is the base URL for the Keyfactor Identity Provider REST API and has an expected value of: `https://<host>/admin/realms/<realm_name>` This parameter is required.

Table 432: Identity Provider Parameter Structure

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the parameter.
Name	A string indicating the short reference name for the parameter (e.g. NameClaimType).

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>DisplayName</td><td>A string indicating the display name for the parameter (e.g. Name Claim Type).</td></tr><tr><td>Required</td><td>A Boolean indicating whether the parameter is required (true) or not (false).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 - String 2 - Secret 3 - Boolean </td></tr><tr><td>Value</td><td>A string indicating the value set for the parameter, for parameters of type 1 or 3.</td></tr><tr><td>SecretValue</td><td>A string indicating the value set for the parameter, for parameters of type 2. Due to its sensitive nature, this value is not returned in responses.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.19.2 PUT Identity Providers ID

The PUT /Identity/Providers/{id} method is used to update an identity provider in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the identity provider.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/identity_providers/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected

data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 433: PUT Identity Providers {id} Input Parameters

Name	In	Description
Id	Path	**Required**. A string containing the Keyfactor Command reference GUID for the identity provider.
AuthenticationScheme	Body	**Required**. A string indicating the authentication scheme for the identity provider.
DisplayName	Body	**Required**. A string indicating the display name for the identity provider.
TypeId	Body	**Required**. A string indicating the Keyfactor Command reference GUID for the type of identity provider. Possible values are: • DFB94650-E4EB-402A-B807-4F3CC91F712D (Active Directory) • F96B6464-11B7-4499-BEA7-B5AA6BA1571D (Generic—select this for Keyfactor Identity Provider) • 5AA04122-CD7C-48BA-AC11-F39E30AE8720 (Auth0)
Parameters	Body	**Required**. An array of objects containing information for each parameter set for the identity provider. Each parameter (Table 434: Identity Provider Parameters) contains the data shown in Table 435: Identity Provider Parameter Structure. Each parameter (Table 434: Identity Provider Parameters) contains the data shown in Table 435: Identity Provider Parameter Structure. *Table 434: Identity Provider Parameters* **Name** / **Type** / **Example** / Description Admin Querying Client Id / 1 - String / Command-API-Query / The client ID of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). Keyfactor recommends that you use a different client for this purpose than the client used for the main connection from Keyfactor Command to the identity provider (see *Client*

Name	In	Description

Name	Type	Example	Description
			Id). This parameter is required.
Admin Querying Client Secret	1 - String		The client secret of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). This parameter is required.
OIDC Audience	1 - String	Command-OIDC-Client	The audience value for the identity provider. For Keyfactor Identity Provider, this should be set to the same value as the *Client Id*. For example: `Command-OIDC-Client` This parameter is required.
Auth0 API URL	1 - String		The unique identifier defined in Auth0 or a similar identity provider for the API. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Authority	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor	The issuer/authority endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the

Name	In	Description

Name	Type	Example	Description
			OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required. **Tip:** When you add or update an identity provider, the provider's discovery document is validated based on this authority URL. The discovery document is also validated periodically in the background. The following are validated: • That the discovery document is reachable using the Authority value provided and can be parsed into a valid discovery document. • That the Authority URL matches the Issuer returned in the discovery

Name	In	Description

Name	Type	Example	Description

document.

- That all the URLs on the discovery document are using HTTPS.
- That the JSONWebKeySetUri value is included on the discovery document.
- That any endpoint configuration values (Authorization Endpoint, Token Endpoint, UserInfo Endpoint, JSONWebKeySetUri) that have been saved or are being saved match—including case—the values returned in the discovery document. The UserInfo Endpoint is not a required configuration field, but if a value is provided, it must match what's in the discovery document.

If any of these validation tests fail, any identity provider changes in process

Name	In	Description

Name	Type	Example	Description
			will not be saved and an error will be displayed or logged.
Authorization Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/-protocol/openid-connect/auth	The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Client Id	1 - String	Command-OIDC-Client	The ID of the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, this should be: `Command-OIDC-Client` For more information, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on

Name	In	Description

Name	Type	Example	Description
			page 2716). This parameter is required.
Client Secret	2 - Secret		The secret for the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716 for help locating this. It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information.

Name	In	Description

Name	Type	Example	Description

Value	Description
SecretValue	A string containing the secret. This parameter is used when PAM is not used as the storage location.
Parameters	An object indicating the parameters to supply for PAM authentication. These will vary depending on the PAM provider.
Provider	A string indicating the ID of the PAM provider. Use the *GET /PamProviders*

Name	In	Description

Name	Type	Example	Description

Value	Description
	method (see GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the ID.

For example, a username stored as a Keyfactor secret will look like:

```
{
   "SecretValue":
"KEYEXAMPLE\svc_MySer-
viceName"
}
```

For example, a password stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "MySu-
perSecretPassword"
}
```

A secret stored as a CyberArk PAM secret will

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898 and the Safe, Folder, and Object reference the information in the CyberArk safe needed for this record):

<pre>{
   "Provider": "1",
   "Parameters":{

"Safe":"MySafeName",

"Folder":"
MyFolderName",
      "Object":"MyOb-
jectName"
   }
}</pre>
A secret stored as a Delinea PAM secret will look like (where the Provider value—2 in this example—is the Id value from GET PAM Providers on page 1898 and the SecretId and SecretFieldName contain the information created in the Delinea secret server for this purpose):

<pre>{
   "Provider": "2",
   "Parameters":{
      "SecretId":"MyId"

"SecretFieldName":"MyRe-</pre></td></tr>
</table>
</td></tr>
</table>

Name	In	Description

		<table><tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr><tr><td></td><td></td><td></td><td><pre>ferenceName"
		  }
		}</pre>
		This parameter is required.</td></tr><tr><td>Discovery Document Endpoint</td><td>1 - String</td><td>https://my-keyidp-server.keyexample.com/realms/Keyfactor/.well-known/openid-configuration</td><td>The discovery URL for the identity provider.
		For Keyfactor Identity Provider, this is the link to the OpenID Endpoint Configuration page, which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). Populate this value and click Fetch to populate the remainder of the fields in this section, if desired.
		If you opt not to populate this field or if the discovery document does not return a valid response, the remainder of the fields in this section of the configuration will need to be configured manually. This value is not stored in the database.</td></tr><tr><td>Fallback Unique Claim Type</td><td>1 - String</td><td>cid</td><td>A backup value used to reference the type of claim used for users in the identity provider in case the primary referenced name does not contain a value.
		This parameter is required.</td></tr></table>

Name	In	Description

Name	Type	Example	Description
JSON Web Key Set Uri	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/-protocol/openid-connect/certs	The JWKS (JSON Web Key Set) URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Name Claim Type	1 - String	preferred_username	The name used to reference the type of user claim for the identity provider. For Keyfactor Identity Provider, this should be: `preferred_username` This parameter is required.
Role Claim Type	1 - String	groups	The value used to reference the type of group claim for the identity provider. For Keyfactor Identity Provider, this should be: `groups`

Name	In	Description

Name	Type	Example	Description
			This parameter is required.
OIDC Scope	1 - String		One or more scopes that are requested during the OIDC protocol when Keyfactor Command is the relying party. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Sign Out URL	1 - String	https://my-auth0-instance.us.auth0.com/oidc/logout	The signout URL for the identity provider. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Timeout	1 - String	60	The number of seconds a request to the identity provider is allowed to process before timing out with an error.
Token Audience	1 - String		An audience value to be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. This value is not used for Keyfactor Identity Provider.
Token Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/-	The token endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included

Name	In	Description

Name	Type	Example	Description
		protocol/openid-connect/token	among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Token Scope	1 - String		One or more scopes that should be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Unique Claim Type	1 - String	sub	The value used to reference the type of claim used for users in the identity provider. For Keyfactor Identity Provider, this should be (for subject): `sub` This parameter is required.

Name	In	Description

Name	Type	Example	Description
User Info Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/-protocol/openid-connect/certs	The user info endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct.
User Query Endpoint	1 - String	https://my-keyidp-server.keyexample.com/admin/realms/Keyfactor	The user query endpoint URL for the identity provider. For Keyfactor Identity Provider, this is the base URL for the Keyfactor Identity Provider REST API and has an expected value of: `https://<host>/admin/realms/<realm_name>` This parameter is required.

Table 435: Identity Provider Parameter Structure

Name	Description
Id	An integer indicating the Keyfactor Command reference

Name	In	Description

Name	Description
	ID for the parameter.
Name	A string indicating the short reference name for the parameter (e.g. NameClaimType).
DisplayName	A string indicating the display name for the parameter (e.g. Name Claim Type).
Required	A Boolean indicating whether the parameter is required (true) or not (false).
DataType	An integer indicating the data type for the parameter. Possible values are: • 1 - String • 2 - Secret • 3 - Boolean
Value	A string indicating the value set for the parameter, for parameters of type 1 or 3.
SecretValue	A string indicating the value set for the parameter, for parameters of type 2. Due to its sensitive nature, this value is not returned in responses.

Table 436: PUT Identity Providers {id} Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the identity provider.
Authentic-ationScheme	A string indicating the authentication scheme for the identity provider.
DisplayName	A string indicating the display name for the identity provider.
TypeId	A string indicating the Keyfactor Command reference GUID for the type of identity provider. Possible values are: • DFB94650-E4EB-402A-B807-4F3CC91F712D (Active Directory) • F96B6464-11B7-4499-BEA7-B5AA6BA1571D (Generic—select this for Keyfactor Identity Provider) • 5AA04122-CD7C-48BA-AC11-F39E30AE8720 (Auth0)
Parameters	An array of objects containing information for each parameter set for the identity provider. Each parameter (Table 434: Identity Provider Parameters) contains the data shown in Table 435: Identity Provider Parameter Structure. Each parameter (Table 434: Identity Provider Parameters) contains the data shown in Table 435: Identity Provider Parameter Structure.

Table 437: Identity Provider Parameters

Name	Type	Example	Description
Admin Querying Client Id	1 - String	Command-API-Query	The client ID of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). Keyfactor recommends that you use a different client for this purpose than the client used for the main connection from Keyfactor Command to the identity provider (see *Client Id*). This parameter is required.

Name	Description

Name	Type	Example	Description
Admin Querying Client Secret	1 - String		The client secret of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). This parameter is required.
OIDC Audience	1 - String	Command-OIDC-Client	The audience value for the identity provider. For Keyfactor Identity Provider, this should be set to the same value as the *Client Id*. For example: `Command-OIDC-Client` This parameter is required.
Auth0 API URL	1 - String		The unique identifier defined in Auth0 or a similar identity provider for the API. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Authority	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor	The issuer/authority endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and

Name	Description

Name	Type	Example	Description
			Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required. **Tip:** When you add or update an identity provider, the provider's discovery document is validated based on this authority URL. The discovery document is also validated periodically in the background. The following are validated: • That the discovery document is reachable using the Authority value provided and can be parsed into a valid discovery document. • That the Authority URL matches the Issuer returned in the discovery document. • That all the URLs on the discovery document are using HTTPS. • That the JSONWebKeySetUri value is

Name	Description

Name	Type	Example	Description
			included on the discovery document. • That any endpoint configuration values (Authorization Endpoint, Token Endpoint, UserInfo Endpoint, JSONWebKeySetUri) that have been saved or are being saved match—including case—the values returned in the discovery document. The UserInfo Endpoint is not a required configuration field, but if a value is provided, it must match what's in the discovery document. If any of these validation tests fail, any identity provider changes in process will not be saved and an error will be displayed or logged.
Authorization Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/auth	The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can

Name	Description

Name	Type	Example	Description
			be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Client Id	1 - String	Command-OIDC-Client	The ID of the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, this should be: `Command-OIDC-Client` For more information, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). This parameter is required.
Client Secret	2 - Secret		The secret for the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716 for help locating this. It is automatically returned by the *Discovery Document*

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
Endpoint Fetch step. Review it to confirm that it appears correct.
Supported methods to store secret information are:
• Store the secret information in the Keyfactor secrets table.
A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.
• Load the secret information from a PAM provider.
See <a>Privileged Access Management (PAM) on page 742 for more information.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the secret. This parameter is used when PAM is not used as the storage location.</td></tr>
<tr><td>Parameters</td><td>An object</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>indicating the parameters to supply for PAM authentication. These will vary depending on the PAM provider.</td></tr>
<tr><td>Provider</td><td>A string indicating the ID of the PAM provider.
Use the GET /PamProviders method (see <a>GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the ID.</td></tr>
</table>

For example, a username stored as a Keyfactor secret will look like:</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description

```
{
   "SecretValue":
"KEYEXAMPLE\svc_MySer-
viceName"
}
```

For example, a password stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "MySu-
perSecretPassword"
}
```

A secret stored as a CyberArk PAM secret will look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898 and the Safe, Folder, and Object reference the information in the CyberArk safe needed for this record):

```
{
   "Provider": "1",
   "Parameters":{
      "Safe":"MySafeName",
      "Folder":"MyFolder-
Name",
      "Object":"MyOb-
jectName"
   }
}
```

A secret stored as a Delinea PAM secret will look like (where the Provider value—2 in this

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>example—is the Id value from GET PAM Providers on page 1898 and the SecretId and SecretFieldName contain the information created in the Delinea secret server for this purpose):

<pre>{
   "Provider": "2",
   "Parameters":{
      "SecretId":"MyId"

"SecretFieldName":"MyReferenceName"
   }
}</pre>
This parameter is required.</td></tr>
<tr><td>Discovery Document Endpoint</td><td>1 - String</td><td>https://my-keyidp-server.keyexample.com/realms/Keyfactor/.well-known/openid-configuration</td><td>The discovery URL for the identity provider.
For Keyfactor Identity Provider, this is the link to the OpenID Endpoint Configuration page, which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). Populate this value and click Fetch to populate the remainder of the fields in this section, if desired.
If you opt not to populate this field or if the discovery document does not return a valid response, the remainder of the fields in this section of the configuration will need to be</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description
			configured manually. This value is not stored in the database.
Fallback Unique Claim Type	1 - String	cid	A backup value used to reference the type of claim used for users in the identity provider in case the primary referenced name does not contain a value. This parameter is required.
JSON Web Key Set Uri	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/certs	The JWKS (JSON Web Key Set) URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Name Claim Type	1 - String	preferred_username	The name used to reference the type of user claim for the identity provider. For Keyfactor Identity Provider, this should be: `preferred_username` This parameter is required.
Role	1 -	groups	The value used to reference the

Name	Description

Name	Type	Example	Description
Claim Type	String		type of group claim for the identity provider. For Keyfactor Identity Provider, this should be: `groups` This parameter is required.
OIDC Scope	1 - String		One or more scopes that are requested during the OIDC protocol when Keyfactor Command is the relying party. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Sign Out URL	1 - String	https://my-auth0-instance.us.auth0.com/oidc/logout	The signout URL for the identity provider. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Timeout	1 - String	60	The number of seconds a request to the identity provider is allowed to process before timing out with an error.
Token Audience	1 - String		An audience value to be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. This value is not used for Keyfactor Identity Provider.
Token	1 -	https://my-keyidp-serv-	The token endpoint URL for the

Name	Description

Name	Type	Example	Description
Endpoint	String	er.keyexample.com/realms/Keyfactor/protocol/openid-connect/token	identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Token Scope	1 - String		One or more scopes that should be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Unique Claim Type	1 - String	sub	The value used to reference the type of claim used for users in the identity provider. For Keyfactor Identity Provider, this should be (for subject): `sub` This parameter is required.
User Info	1 -	https://my-keyidp-serv-	The user info endpoint URL for

Name	Description

Name	Type	Example	Description
Endpoint	String	er.keyexample.com /realms/Keyfactor/protocol/openid-connect/certs	the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct.
User Query Endpoint	1 - String	https://my-keyidp-server.keyexample.com /admin/realms/Keyfactor	The user query endpoint URL for the identity provider. For Keyfactor Identity Provider, this is the base URL for the Keyfactor Identity Provider REST API and has an expected value of: `https://<host>/admin/realms/<realm_name>` This parameter is required.

Table 438: Identity Provider Parameter Structure

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the parameter.
Name	A string indicating the short reference name for the parameter (e.g. NameClaimType).

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>DisplayName</td><td>A string indicating the display name for the parameter (e.g. Name Claim Type).</td></tr><tr><td>Required</td><td>A Boolean indicating whether the parameter is required (true) or not (false).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 - String 2 - Secret 3 - Boolean </td></tr><tr><td>Value</td><td>A string indicating the value set for the parameter, for parameters of type 1 or 3.</td></tr><tr><td>SecretValue</td><td>A string indicating the value set for the parameter, for parameters of type 2. Due to its sensitive nature, this value is not returned in responses.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.19.3 GET Identity Providers

The GET /Identity/Providers method is used to return the list of security identity providers configured in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the identity providers.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
SecuritySettings: *Read*

Table 439: GET Identity Providers Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Identity Provider Search Feature on page 763. The query fields supported for this endpoint are: • DisplayName • Name • Private • ProviderTypes
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Provider*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 440: GET Identity Provider Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the identity provider.
AuthenticationScheme	A string indicating the authentication scheme for the identity provider.
DisplayName	A string indicating the display name for the identity provider.
TypeId	A string indicating the Keyfactor Command reference GUID for the type of identity provider. Possible values are: • DFB94650-E4EB-402A-B807-4F3CC91F712D (Active Directory) • F96B6464-11B7-4499-BEA7-B5AA6BA1571D (Generic—select this for Keyfactor Identity Provider) • 5AA04122-CD7C-48BA-AC11-F39E30AE8720 (Auth0)
Parameters	An array of objects containing information for each parameter set for the identity provider. Each parameter (Table 441: Identity Provider Parameters) contains the data shown in Table 442: Identity Provider Parameter Structure. Each parameter (Table 441: Identity Provider Parameters) contains the data shown in Table 442: Identity Provider Parameter Structure.

Table 441: Identity Provider Parameters

Name	Type	Example	Description
Admin Querying Client Id	1 - String	Command-API-Query	The client ID of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). Keyfactor recommends that you use a different client for this purpose than the client used for the main connection from Keyfactor Command to the identity provider (see *Client Id*). This parameter is required.

Name	Description

Name	Type	Example	Description
Admin Querying Client Secret	1 - String		The client secret of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). This parameter is required.
OIDC Audience	1 - String	Command-OIDC-Client	The audience value for the identity provider. For Keyfactor Identity Provider, this should be set to the same value as the *Client Id*. For example: `Command-OIDC-Client` This parameter is required.
Auth0 API URL	1 - String		The unique identifier defined in Auth0 or a similar identity provider for the API. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Authority	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor	The issuer/authority endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and

Name	Description

Name	Type	Example	Description
			Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required. **Tip:** When you add or update an identity provider, the provider's discovery document is validated based on this authority URL. The discovery document is also validated periodically in the background. The following are validated: • That the discovery document is reachable using the Authority value provided and can be parsed into a valid discovery document. • That the Authority URL matches the Issuer returned in the discovery document. • That all the URLs on the discovery document are using HTTPS. • That the JSONWebKeySetUri value is

Name	Description

Name	Type	Example	Description
			included on the discovery document. • That any endpoint configuration values (Authorization Endpoint, Token Endpoint, UserInfo Endpoint, JSONWebKeySetUri) that have been saved or are being saved match—including case—the values returned in the discovery document. The UserInfo Endpoint is not a required configuration field, but if a value is provided, it must match what's in the discovery document. If any of these validation tests fail, any identity provider changes in process will not be saved and an error will be displayed or logged.
Authorization Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/auth	The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can

Name	Description

Name	Type	Example	Description
			be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Client Id	1 - String	Command-OIDC-Client	The ID of the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, this should be: `Command-OIDC-Client` For more information, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). This parameter is required.
Client Secret	2 - Secret		The secret for the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716 for help locating this. It is automatically returned by the *Discovery Document*

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
Endpoint Fetch step. Review it to confirm that it appears correct.
Supported methods to store secret information are:
• Store the secret information in the Keyfactor secrets table.
A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.
• Load the secret information from a PAM provider.
See <a>Privileged Access Management (PAM) on page 742 for more information.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the secret. This parameter is used when PAM is not used as the storage location.</td></tr>
<tr><td>Parameters</td><td>An object</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>indicating the parameters to supply for PAM authentication. These will vary depending on the PAM provider.</td></tr>
<tr><td>Provider</td><td>A string indicating the ID of the PAM provider.
Use the GET /PamProviders method (see GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the ID.</td></tr>
</table>
For example, a username stored as a Keyfactor secret will look like:
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description

```
{
   "SecretValue":
"KEYEXAMPLE\svc_MySer-
viceName"
}
```

For example, a password stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "MySu-
perSecretPassword"
}
```

A secret stored as a CyberArk PAM secret will look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1898 and the Safe, Folder, and Object reference the information in the CyberArk safe needed for this record):

```
{
   "Provider": "1",
   "Parameters":{
      "Safe":"MySafeName",
      "Folder":"MyFolder-
Name",
      "Object":"MyOb-
jectName"
   }
}
```

A secret stored as a Delinea PAM secret will look like (where the Provider value—2 in this

Name	Description

Name	Type	Example	Description
			example—is the Id value from GET PAM Providers on page 1898 and the SecretId and SecretFieldName contain the information created in the Delinea secret server for this purpose): `{ "Provider": "2", "Parameters":{ "SecretId":"MyId" "SecretFieldName":"MyReferenceName" } }` This parameter is required.
Discovery Document Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/.well-known/openid-configuration	The discovery URL for the identity provider. For Keyfactor Identity Provider, this is the link to the OpenID Endpoint Configuration page, which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). Populate this value and click **Fetch** to populate the remainder of the fields in this section, if desired. If you opt not to populate this field or if the discovery document does not return a valid response, the remainder of the fields in this section of the configuration will need to be

Name	Description

Name	Type	Example	Description
			configured manually. This value is not stored in the database.
Fallback Unique Claim Type	1 - String	cid	A backup value used to reference the type of claim used for users in the identity provider in case the primary referenced name does not contain a value. This parameter is required.
JSON Web Key Set Uri	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/certs	The JWKS (JSON Web Key Set) URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Name Claim Type	1 - String	preferred_username	The name used to reference the type of user claim for the identity provider. For Keyfactor Identity Provider, this should be: `preferred_username` This parameter is required.
Role	1 -	groups	The value used to reference the

Name	Description

Name	Type	Example	Description
Claim Type	String		type of group claim for the identity provider. For Keyfactor Identity Provider, this should be: `groups` This parameter is required.
OIDC Scope	1 - String		One or more scopes that are requested during the OIDC protocol when Keyfactor Command is the relying party. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Sign Out URL	1 - String	https://my-auth0-instance.us.auth0.com/oidc/logout	The signout URL for the identity provider. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Timeout	1 - String	60	The number of seconds a request to the identity provider is allowed to process before timing out with an error.
Token Audience	1 - String		An audience value to be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. This value is not used for Keyfactor Identity Provider.
Token	1 -	https://my-keyidp-serv-	The token endpoint URL for the

Name	Description

Name	Type	Example	Description
Endpoint	String	er.keyexample.com/realms/Keyfactor/protocol/openid-connect/token	identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Token Scope	1 - String		One or more scopes that should be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Unique Claim Type	1 - String	sub	The value used to reference the type of claim used for users in the identity provider. For Keyfactor Identity Provider, this should be (for subject): `sub` This parameter is required.
User Info	1 -	https://my-keyidp-serv-	The user info endpoint URL for

Name	Description

Name	Type	Example	Description
Endpoint	String	er.keyexample.com /realms/Keyfactor/protocol/openid-connect/certs	the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct.
User Query Endpoint	1 - String	https://my-keyidp-server.keyexample.com /admin/realms/Keyfactor	The user query endpoint URL for the identity provider. For Keyfactor Identity Provider, this is the base URL for the Keyfactor Identity Provider REST API and has an expected value of: `https://<host>/admin/realms/<realm_name>` This parameter is required.

Table 442: Identity Provider Parameter Structure

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the parameter.
Name	A string indicating the short reference name for the parameter (e.g. NameClaimType).

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>DisplayName</td><td>A string indicating the display name for the parameter (e.g. Name Claim Type).</td></tr><tr><td>Required</td><td>A Boolean indicating whether the parameter is required (true) or not (false).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 - String 2 - Secret 3 - Boolean </td></tr><tr><td>Value</td><td>A string indicating the value set for the parameter, for parameters of type 1 or 3.</td></tr><tr><td>SecretValue</td><td>A string indicating the value set for the parameter, for parameters of type 2. Due to its sensitive nature, this value is not returned in responses.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.19.4 GET Identity Providers Types

The GET /Identity/Providers/Types method is used to list the types of identity providers defined in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the identity provider types and their type parameters. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/identity_providers/read/

Table 443: GET Identity Providers Types Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the identity provider type.
Name	A string containing the name for the identity provider type.
TypeParameters	An object containing information about the identity provider types. Identity provider type information includes:

Name	Description
Id	An integer containing the Keyfactor Command identifier for the identity provider type.
Name	A string containing the short reference name for the identity provider type.
DisplayName	A string containing the display name for the identity provider type.
DataType	An integer indicating the data type of the identity provider type. Possible values are: • 1 - String • 2 - Secret • 3 - Boolean
Required	A Boolean that indicates whether the identity provider type has been marked as required (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.20 License

The License component of the Keyfactor API is primarily intended to view the current license through the API with the GET /License Method.

Table 444: License Endpoint

Endpoint	Method	Description	Link
/	GET	Returns the current license.	GET License below

3.6.20.1 GET License

The GET /License method is used to view the current license. This method returns HTTP 200 OK on a success with the license details. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849). For more information regarding licensing, see Licensing on page 768.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
SystemSettings > Read

Table 445: GET License Response Data

Name	Description
KeyfactorVersion	A string indicating the Keyfactor Command version number in the format: `majorversion.incrementalversion.patchnumber`
LicenseData	An object containing your Keyfactor customer information. License data details are: **Name** / **Description** LicenseId — A string indicating the internal reference GUID of your Keyfactor license. Customer — An object containing identifying information about your organization. **Name** / **Description** Name — A string containing your company name as per your Keyfactor account. Id — An integer containing your Keyfactor account number.
IssuedDate	A string indicating the valid issue date of the license, in UTC.
ExpirationDate	A string indicating the valid expiration date of the license, in UTC.
LicensedProducts	An array of objects containing details of the products and features included in the license. License product and feature details are: **Name** / **Description** ProductId — A string indicating the Keyfactor Command product GUID for the product(s) included in the license. DisplayName — A string indicating the name of the licensed product. For Keyfactor Command, this is “Certificate Management System”. MajorRev — A string indicating the valid major release

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>version of the license.</td></tr><tr><td>MinorRev</td><td>A string indicating the valid incremental release version of the license.</td></tr><tr><td>LicensedFeatures</td><td>An array of objects containing the Keyfactor Command features included in the license.<table><tr><th>Name</th><th>Description</th></tr><tr><td>FeatureID</td><td>A string indicating the ID code of feature.</td></tr><tr><td>DisplayName</td><td>A string indicating the name of the feature as displayed on the license page in the Management Portal.</td></tr><tr><td>Enabled</td><td>A Boolean that indicates whether the feature is enabled (true) or not (false).</td></tr><tr><td>Quantity</td><td>An integer indicating one of: - How many of the elements you are licensed for. - For those features which have no licensing limits, null . Unlimited is indicated by 999999999.</td></tr><tr><td>ExpirationDate</td><td>This field is unused and will always return null.</td></tr></table></td></tr></table>

Name	Description
	Tip: Currently there is only one licensed product offered, which is Keyfactor Command.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.21 MacEnrollment

The MacEnrollment component of the Keyfactor API includes methods to edit and retrieve the configuration for Mac auto-enrollment.

Table 446: MacEnrollment Endpoints

Endpoint	Method	Description	Link
/	GET	Returns the current Mac auto-enrollment configuration.	GET MacEnrollment below
/	PUTT	Updates the Mac auto-enrollment configuration.	PUT MacEnrollment on the next page

3.6.21.1 GET MacEnrollment

The GET /MacEnrollment method is used to retrieve details for the Mac Auto-Enrollment configuration. This method returns HTTP 200 OK on a success with the Mac Auto-Enrollment configuration details. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/mac/auto-enrollment/management/read/

Table 447: GET MacEnrollment Response Data

Name	Description
Id	An integer indicating the Keyfactor Command referenced ID of the Mac auto-enrollment configuration.
Enabled	An Boolean indicating whether Mac auto-enrollment is configured in the environment (true) or not (false).
Interval	An integer indicating the frequency with which the Mac auto-enrollment agent should check to see if there are new certificates for which to enroll.
UseMetadata	A Boolean indicating whether to automatically associate data in a custom metadata field with an auto-enrolled Mac certificate (true) or not (false). See Certificate Metadata on page 710 for more information about metadata fields.
MetadataField	A string indicating the name of the metadata field to populate for the certificate, if *UseMetadata* is *true*.
MetadataValue	A string indicating the value to populate for the metadata field, if *UseMetadata* is *true*. This may be either a static value (e.g. a fixed string that indicates this certificate was acquired as a result of an auto-enrollment on a Mac), or a variable retrieved from the Mac. In the current version of the agent, only the Mac serial number is available.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.21.2 PUT MacEnrollment

The PUT /MacEnrollment method is used to update the existing Mac Auto-Enrollment configuration. This method returns HTTP 200 OK on a success with the Mac Auto-Enrollment configuration details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/mac/auto-enrollment/management/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected

data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 448: PUT MacEnrollment input Parameters

Name	In	Description
Id	Body	An integer indicating the Keyfactor Command referenced ID of the Mac auto-enrollment configuration.
Enabled	Body	An Boolean indicating whether Mac auto-enrollment is configured in the environment (true) or not (false).
Interval	Body	An integer indicating the frequency with which the Mac auto-enrollment agent should check to see if there are new certificates for which to enroll.
UseMetadata	Body	A Boolean indicating whether to automatically associate data in a custom metadata field with an auto-enrolled Mac certificate (true) or not (false). See for more information about metadata fields.
MetadataField	Body	A string indicating the name of the metadata field to populate for the certificate, if *UseMetadata* is *true*.
MetadataValue	Body	A string indicating the value to populate for the metadata field, if *UseMetadata* is *true*. This may be either a static value (e.g. a fixed string that indicates this certificate was acquired as a result of an auto-enrollment on a Mac), or a variable retrieved from the Mac. In the current version of the agent, only the Mac serial number is available.

Table 449: PUT MacEnrollment Response Data

Name	Description
Id	An integer indicating the Keyfactor Command referenced ID of the Mac auto-enrollment configuration.
Enabled	An Boolean indicating whether Mac auto-enrollment is configured in the environment (true) or not (false).
Interval	An integer indicating the frequency with which the Mac auto-enrollment agent should check to see if there are new certificates for which to enroll.
UseMetadata	A Boolean indicating whether to automatically associate data in a custom metadata field with an auto-enrolled Mac certificate (true) or not (false). See Certificate Metadata on page 710 for more information about metadata fields.
MetadataField	A string indicating the name of the metadata field to populate for the certificate, if *UseMetadata* is *true*.
MetadataValue	A string indicating the value to populate for the metadata field, if *UseMetadata* is *true*. This may be either a static value (e.g. a fixed string that indicates this certificate was acquired as a result of an auto-enrollment on a Mac), or a variable retrieved from the Mac. In the current version of the agent, only the Mac serial number is available.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.22 MetadataFields

MetadataFields contains definitions for metadata that can be associated with certificates in Keyfactor Command.

Table 450: MetadataFields Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes an existing metadata field.	DELETE Metadata Fields ID on the next page
/{id}	GET	Returns detailed information for the specified	GET Metadata

Endpoint	Method	Description	Link
		metadata field.	Fields ID on page 1785
/{name}	GET	Returns detailed information for the specified metadata field.	GET Metadata Fields Name on page 1788
/{id}/InUse	GET	Returns a Boolean stating whether the metadata type is associated with a certificate.	GET Metadata Fields ID In Use on page 1792
/	DELETE	Deletes multiple metadata fields specified in the request body.	DELETE Metadata Fields on page 1793
/	GET	Returns all metadata field types with paging (number of pages to return and number of results per page) options.	GET Metadata Fields on page 1794
/	POST	Creates a new metadata field using values supplied in the request body.	POST Metadata Fields on page 1798
/	PUT	Updates an existing metadata field using values supplied in the request body.	PUT Metadata Fields on page 1804

3.6.22.1 DELETE Metadata Fields ID

The DELETE /MetadataFields/{id} method is used to delete a metadata field by ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/metadata/types/modify/

Table 451: DELETE Metadata Fields {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID for the metadata field to be deleted. Use the *GET /MetadataFields* method (see GET Metadata Fields on page 1794) to retrieve a list of all the metadata fields to determine the metadata field's ID.
Force	Query	A Boolean that sets whether to force deletion of the metadata field even if it is in use by one or more certificates (true) or not (false). The default is *false*. Use the *GET /MetadataFields/{id}/InUse* method (see GET Metadata Fields ID In Use on page 1792) to determine whether a metadata field is in use.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.22.2 GET Metadata Fields ID

The GET /MetadataFields/{id} method is used to return details for the metadata field with a specified unique ID. This method returns HTTP 200 OK on a success with details for the requested metadata field.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/metadata/types/read/

Table 452: GET Metadata Fields {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the metadata field. Use the *GET /MetadataFields* method (see GET Metadata Fields on page 1794) to retrieve a list of all the metadata fields to determine the metadata field's ID.

Table 453: GET Metadata Fields {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</td><td>An integer indicating the Keyfactor Command reference ID for the metadata field. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the metadata field. This name appears in interfaces where you can use metadata, such as certificate details dialogs, alert dialogs, certificate imports and certificate requests. Once this field has a value associated with it for at least one certificate, you cannot change this name. The metadata name field cannot contain spaces; dashes and underscores are supported.</td></tr>
<tr><td>Description</td><td>A string indicating the description for the metadata field.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type of the metadata field. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>String</td></tr><tr><td>2</td><td>Integer</td></tr><tr><td>3</td><td>Date</td></tr><tr><td>4</td><td>Boolean</td></tr><tr><td>5</td><td>Multiple Choice</td></tr><tr><td>6</td><td>Big Text</td></tr></table></td></tr>
<tr><td>Hint</td><td>A string indicating a short hint for the metadata field. This hint appears in unpopulated metadata string, integer, big text and date fields on editing interfaces to provide the user with a clue as to what type of data should be entered in the field.
This field is only supported for metadata fields with data types string, integer, date or big text.</td></tr>
<tr><td>Validation</td><td>A string containing a regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:
<code>^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$</code>
This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>This field is only supported for metadata fields with data type string.

Tip: If a template specific option is set for a given metadata field, that takes precedence over the global options. The template-specific regular expression will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Enrollment</td><td>An integer indicating how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr>
<tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr>
<tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr>
</table>

Tip: If a template-specific handling is set for a given metadata field, it takes precedence over this global setting. The template-specific handling will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Message</td><td>A string containing a message to present when a user enters information in a metadata field that does not match the specified regular expression (Validation field).

Tip: If a template-specific regular expression message is set for a given metadata field, it takes precedence over this global regular expression message. The template-specific message will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Options</td><td>An array containing a comma separated list of values that should appear in the field dropdown for multiple choice fields.
This field is only supported for metadata fields with data type multiple choice.</td></tr>
</table>

Name	Description
	Tip: If a template-specific options are set for a given metadata field, these takes precedence over these global options. The template-specific options will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
DefaultValue	A string containing a default value with which to pre-populate the metadata field for new certificate requests made using PFX or CSR enrollment. Data type of Email will accept a comma separated list of email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, or *multiple choice*. **Tip:** If a template-specific default is set for a given metadata field, it takes precedence over this global default value. The template-specific default will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
AllowAPI	This is considered deprecated and may be removed in a future release.
ExplicitUpdate	This is considered deprecated and may be removed in a future release.
DisplayOrder	An integer indicating the order in which the metadata field should be displayed on pages where the metadata fields are displayed (e.g. PFX enrollment, certificate details).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.22.3 GET Metadata Fields Name

The GET /MetadataFields/{name} method is used to return details for the metadata field with the specified unique name. This method returns HTTP 200 OK on a success with details for the requested metadata field.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/metadata/types/read/

Table 454: GET Metadata Fields {name} Input Parameters

Name	In	Description
name	Path	**Required**. A string that indicates the name of the metadata field. This value is not case sensitive.

Table 455: GET Metadata Fields {name} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</td><td>An integer indicating the Keyfactor Command reference ID for the metadata field. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the metadata field. This name appears in interfaces where you can use metadata, such as certificate details dialogs, alert dialogs, certificate imports and certificate requests. Once this field has a value associated with it for at least one certificate, you cannot change this name. The metadata name field cannot contain spaces; dashes and underscores are supported.</td></tr>
<tr><td>Description</td><td>A string indicating the description for the metadata field.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type of the metadata field. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>String</td></tr><tr><td>2</td><td>Integer</td></tr><tr><td>3</td><td>Date</td></tr><tr><td>4</td><td>Boolean</td></tr><tr><td>5</td><td>Multiple Choice</td></tr><tr><td>6</td><td>Big Text</td></tr></table></td></tr>
<tr><td>Hint</td><td>A string indicating a short hint for the metadata field. This hint appears in unpopulated metadata string, integer, big text and date fields on editing interfaces to provide the user with a clue as to what type of data should be entered in the field.
This field is only supported for metadata fields with data types string, integer, date or big text.</td></tr>
<tr><td>Validation</td><td>A string containing a regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:
<code>^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$</code>
This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>This field is only supported for metadata fields with data type string.

Tip: If a template specific option is set for a given metadata field, that takes precedence over the global options. The template-specific regular expression will be used in PFX and CSR enrollment requests using that template (see <a>GET Templates ID on page 2377).</td></tr>
<tr><td>Enrollment</td><td>An integer indicating how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr>
<tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr>
<tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr>
</table>

Tip: If a template-specific handling is set for a given metadata field, it takes precedence over this global setting. The template-specific handling will be used in PFX and CSR enrollment requests using that template (see <a>GET Templates ID on page 2377).</td></tr>
<tr><td>Message</td><td>A string containing a message to present when a user enters information in a metadata field that does not match the specified regular expression (Validation field).

Tip: If a template-specific regular expression message is set for a given metadata field, it takes precedence over this global regular expression message. The template-specific message will be used in PFX and CSR enrollment requests using that template (see <a>GET Templates ID on page 2377).</td></tr>
<tr><td>Options</td><td>An array containing a comma separated list of values that should appear in the field dropdown for multiple choice fields.
This field is only supported for metadata fields with data type multiple choice.</td></tr>
</table>

Name	Description
	Tip: If a template-specific options are set for a given metadata field, these takes precedence over these global options. The template-specific options will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
DefaultValue	A string containing a default value with which to pre-populate the metadata field for new certificate requests made using PFX or CSR enrollment. Data type of Email will accept a comma separated list of email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, or *multiple choice*. **Tip:** If a template-specific default is set for a given metadata field, it takes precedence over this global default value. The template-specific default will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
AllowAPI	This is considered deprecated and may be removed in a future release.
ExplicitUpdate	This is considered deprecated and may be removed in a future release.
DisplayOrder	An integer indicating the order in which the metadata field should be displayed on pages where the metadata fields are displayed (e.g. PFX enrollment, certificate details).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.22.4 GET Metadata Fields ID In Use

The GET /MetadataFields/{id}/InUse method is used to return a Boolean indicating whether the specified metadata field contains any data for any of the certificates in Keyfactor Command. This is useful to determine before attempting to delete a metadata field. This method returns HTTP 200 OK on a success with a value of true or false.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/metadata/types/read/

Table 456: GET Metadata Fields {id} In Use Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID for the metadata field. Use the *GET /MetadataFields* method (see GET Metadata Fields on the next page) to retrieve a list of all the metadata fields to determine the metadata field's ID.

Table 457: GET Metadata Fields {id} In Use Response Data

Name	Description
	A Boolean that indicates whether the specified metadata field contains data for any certificates within Keyfactor Command (true) or not (false). This value is returned without a parameter name.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.22.5 DELETE Metadata Fields

The DELETE /MetadataFields method is used to delete multiple metadata fields in one request. Delete operations will continue until the entire array of IDs has been processed. Note that metadata fields that are in use for any certificate cannot be deleted unless the force=true parameter is included in the request. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/metadata/types/modify/

Table 458: DELETE Metadata Fields Input Parameters

Name	In	Description
ids	Body	**Required**. An array of integers indicating the Keyfactor Command reference IDs for the metadata fields to be deleted. Use the *GET /MetadataFields* method (see GET Metadata Fields below) to retrieve a list of all the metadata fields to determine the metadata field IDs.
Force	Query	A Boolean that sets whether to force deletion of the metadata fields even if they are in use (true) or not (false). The default is *False*. Use the *GET /MetadataFields/{id}/InUse* method (see GET Metadata Fields ID In Use on page 1792) to determine whether a metadata field is in use.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.22.6 GET Metadata Fields

The GET /MetadataFields method is used to return a list of all metadata fields. This method returns HTTP 200 OK on a success with details for the metadata fields.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/metadata/types/read/

Table 459: GET Metadata Fields Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Logons Search on page 590. The query fields supported for this endpoint are: • Name
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayOrder*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 460: GET Metadata Fields Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</td><td>An integer indicating the Keyfactor Command reference ID for the metadata field. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the metadata field. This name appears in interfaces where you can use metadata, such as certificate details dialogs, alert dialogs, certificate imports and certificate requests. Once this field has a value associated with it for at least one certificate, you cannot change this name. The metadata name field cannot contain spaces; dashes and underscores are supported.</td></tr>
<tr><td>Description</td><td>A string indicating the description for the metadata field.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type of the metadata field. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>String</td></tr><tr><td>2</td><td>Integer</td></tr><tr><td>3</td><td>Date</td></tr><tr><td>4</td><td>Boolean</td></tr><tr><td>5</td><td>Multiple Choice</td></tr><tr><td>6</td><td>Big Text</td></tr></table></td></tr>
<tr><td>Hint</td><td>A string indicating a short hint for the metadata field. This hint appears in unpopulated metadata string, integer, big text and date fields on editing interfaces to provide the user with a clue as to what type of data should be entered in the field.
This field is only supported for metadata fields with data types string, integer, date or big text.</td></tr>
<tr><td>Validation</td><td>A string containing a regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:
<code>^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$</code>
This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>This field is only supported for metadata fields with data type string.

Tip: If a template specific option is set for a given metadata field, that takes precedence over the global options. The template-specific regular expression will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Enrollment</td><td>An integer indicating how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr>
<tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr>
<tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr>
</table>

Tip: If a template-specific handling is set for a given metadata field, it takes precedence over this global setting. The template-specific handling will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Message</td><td>A string containing a message to present when a user enters information in a metadata field that does not match the specified regular expression (Validation field).

Tip: If a template-specific regular expression message is set for a given metadata field, it takes precedence over this global regular expression message. The template-specific message will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Options</td><td>An array containing a comma separated list of values that should appear in the field dropdown for multiple choice fields.
This field is only supported for metadata fields with data type multiple choice.</td></tr>
</table>

Name	Description
	Tip: If a template-specific options are set for a given metadata field, these takes precedence over these global options. The template-specific options will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
DefaultValue	A string containing a default value with which to pre-populate the metadata field for new certificate requests made using PFX or CSR enrollment. Data type of Email will accept a comma separated list of email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, or *multiple choice*. **Tip:** If a template-specific default is set for a given metadata field, it takes precedence over this global default value. The template-specific default will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
AllowAPI	This is considered deprecated and may be removed in a future release.
ExplicitUpdate	This is considered deprecated and may be removed in a future release.
DisplayOrder	An integer indicating the order in which the metadata field should be displayed on pages where the metadata fields are displayed (e.g. PFX enrollment, certificate details).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.22.7 POST Metadata Fields

The POST /MetadataFields method is used to create a new metadata field in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the new metadata field.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/metadata/types/modify/

Table 461: POST Metadata Fields Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the metadata field. This name appears in interfaces where you can use metadata, such as certificate details dialogs, alert dialogs, certificate imports and certificate requests. Once this field has a value associated with it for at least one certificate, you cannot change this name. The metadata name field cannot contain spaces; dashes and underscores are supported.</td></tr>
<tr><td>Description</td><td>Body</td><td>Required. A string indicating the description for the metadata field.</td></tr>
<tr><td>DataType</td><td>Body</td><td>Required. An integer indicating the data type of the metadata field. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td></tr>
<tr><td>2</td><td>Integer</td></tr>
<tr><td>3</td><td>Date</td></tr>
<tr><td>4</td><td>Boolean</td></tr>
<tr><td>5</td><td>Multiple Choice</td></tr>
<tr><td>6</td><td>Big Text</td></tr>
</table></td></tr>
<tr><td>Hint</td><td>Body</td><td>A string indicating a short hint for the metadata field. This hint appears in unpopulated metadata string, integer, big text and date fields on editing interfaces to provide the user with a clue as to what type of data should be entered in the field.
This field is only supported for metadata fields with data types string, integer, date or big text.</td></tr>
<tr><td>Validation</td><td>Body</td><td>A string containing a regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:
<code>^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$</code>
This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, under-</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>scores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.
This field is only supported for metadata fields with data type string.

Tip: If a template specific option is set for a given metadata field, that takes precedence over the global options. The template-specific regular expression will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Enrollment</td><td>Body</td><td>An integer indicating how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr><tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr><tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr></table>
The default is optional.

Tip: If a template-specific handling is set for a given metadata field, it takes precedence over this global setting. The template-specific handling will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Message</td><td>Body</td><td>A string containing a message to present when a user enters information in a metadata field that does not match the specified regular expression (Validation field).</td></tr>
</table>

Name	In	Description
		Tip: If a template-specific regular expression message is set for a given metadata field, it takes precedence over this global regular expression message. The template-specific message will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
Options	Body	An array containing a comma separated list of values that should appear in the field dropdown for multiple choice fields. This field is **required** for metadata fields with data type *multiple choice*. For other data types, it will be ignored. **Tip:** If a template-specific options are set for a given metadata field, these takes precedence over these global options. The template-specific options will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
DefaultValue	Body	A string containing a default value with which to pre-populate the metadata field for new certificate requests made using PFX or CSR enrollment. Data type of Email will accept a comma separated list of email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, or *multiple choice*. **Tip:** If a template-specific default is set for a given metadata field, it takes precedence over this global default value. The template-specific default will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
AllowAPI	Body	This is considered deprecated and may be removed in a future release.
ExplicitUpdate	Body	This is considered deprecated and may be removed in a future release.
DisplayOrder	Body	**Required**. An integer indicating the order in which the metadata field should be displayed on pages where the metadata fields are displayed (e.g. PFX enrollment, certificate details).

Table 462: POST Metadata Fields Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</td><td>An integer indicating the Keyfactor Command reference ID for the metadata field. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the metadata field. This name appears in interfaces where you can use metadata, such as certificate details dialogs, alert dialogs, certificate imports and certificate requests. Once this field has a value associated with it for at least one certificate, you cannot change this name. The metadata name field cannot contain spaces; dashes and underscores are supported.</td></tr>
<tr><td>Description</td><td>A string indicating the description for the metadata field.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type of the metadata field. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>String</td></tr><tr><td>2</td><td>Integer</td></tr><tr><td>3</td><td>Date</td></tr><tr><td>4</td><td>Boolean</td></tr><tr><td>5</td><td>Multiple Choice</td></tr><tr><td>6</td><td>Big Text</td></tr></table></td></tr>
<tr><td>Hint</td><td>A string indicating a short hint for the metadata field. This hint appears in unpopulated metadata string, integer, big text and date fields on editing interfaces to provide the user with a clue as to what type of data should be entered in the field.
This field is only supported for metadata fields with data types string, integer, date or big text.</td></tr>
<tr><td>Validation</td><td>A string containing a regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:
<code>^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$</code>
This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>This field is only supported for metadata fields with data type string.

Tip: If a template specific option is set for a given metadata field, that takes precedence over the global options. The template-specific regular expression will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Enrollment</td><td>An integer indicating how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr>
<tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr>
<tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr>
</table>

Tip: If a template-specific handling is set for a given metadata field, it takes precedence over this global setting. The template-specific handling will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Message</td><td>A string containing a message to present when a user enters information in a metadata field that does not match the specified regular expression (Validation field).

Tip: If a template-specific regular expression message is set for a given metadata field, it takes precedence over this global regular expression message. The template-specific message will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Options</td><td>An array containing a comma separated list of values that should appear in the field dropdown for multiple choice fields.
This field is only supported for metadata fields with data type multiple choice.</td></tr>
</table>

Name	Description
	Tip: If a template-specific options are set for a given metadata field, these takes precedence over these global options. The template-specific options will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
DefaultValue	A string containing a default value with which to pre-populate the metadata field for new certificate requests made using PFX or CSR enrollment. Data type of Email will accept a comma separated list of email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, or *multiple choice*. **Tip:** If a template-specific default is set for a given metadata field, it takes precedence over this global default value. The template-specific default will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
AllowAPI	This is considered deprecated and may be removed in a future release.
ExplicitUpdate	This is considered deprecated and may be removed in a future release.
DisplayOrder	An integer indicating the order in which the metadata field should be displayed on pages where the metadata fields are displayed (e.g. PFX enrollment, certificate details).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.22.8 PUT Metadata Fields

The PUT /MetadataFields method is used to update an existing metadata field in Keyfactor Command. This method returns HTTP 200 OK on a success with details of the updated metadata field.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/metadata/types/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 463: PUT Metadata Fields Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ID</td><td>Body</td><td>Required. An integer indicating the Keyfactor Command reference ID for the metadata field. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the metadata field. This name appears in interfaces where you can use metadata, such as certificate details dialogs, alert dialogs, certificate imports and certificate requests. Once this field has a value associated with it for at least one certificate, you cannot change this name. The metadata name field cannot contain spaces; dashes and underscores are supported.</td></tr>
<tr><td>Description</td><td>Body</td><td>Required. A string indicating the description for the metadata field.</td></tr>
<tr><td>DataType</td><td>Body</td><td>Required. An integer indicating the data type of the metadata field. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td></tr>
<tr><td>2</td><td>Integer</td></tr>
<tr><td>3</td><td>Date</td></tr>
<tr><td>4</td><td>Boolean</td></tr>
<tr><td>5</td><td>Multiple Choice</td></tr>
<tr><td>6</td><td>Big Text</td></tr>
</table></td></tr>
<tr><td>Hint</td><td>Body</td><td>A string indicating a short hint for the metadata field. This hint appears in unpopulated metadata string, integer, big text and date fields on editing interfaces to provide the user with a clue as to what type of data should be entered in the field.
This field is only supported for metadata fields with data types string, integer, date or big text.</td></tr>
<tr><td>Validation</td><td>Body</td><td>A string containing a regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:
<code>^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$</code></td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.
This field is only supported for metadata fields with data type string.

Tip: If a template specific option is set for a given metadata field, that takes precedence over the global options. The template-specific regular expression will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Enrollment</td><td>Body</td><td>An integer indicating how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr><tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr><tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr></table>
The default is optional.

Tip: If a template-specific handling is set for a given metadata field, it takes precedence over this global setting. The template-specific handling will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Message</td><td>Body</td><td>A string containing a message to present when a user enters information</td></tr>
</table>

Name	In	Description
		in a metadata field that does not match the specified regular expression (*Validation* field). **Tip:** If a template-specific regular expression message is set for a given metadata field, it takes precedence over this global regular expression message. The template-specific message will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
Options	Body	An array containing a comma separated list of values that should appear in the field dropdown for multiple choice fields. This field is **required** for metadata fields with data type *multiple choice*. For other data types, it will be ignored. **Tip:** If a template-specific options are set for a given metadata field, these takes precedence over these global options. The template-specific options will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
DefaultValue	Body	A string containing a default value with which to pre-populate the metadata field for new certificate requests made using PFX or CSR enrollment. Data type of Email will accept a comma separated list of email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, or *multiple choice*. **Tip:** If a template-specific default is set for a given metadata field, it takes precedence over this global default value. The template-specific default will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
AllowAPI	Body	This is considered deprecated and may be removed in a future release.
ExplicitUpdate	Body	This is considered deprecated and may be removed in a future release.
DisplayOrder	Body	**Required**. An integer indicating the order in which the metadata field should be displayed on pages where the metadata fields are displayed (e.g. PFX enrollment, certificate details).

Table 464: PUT Metadata Fields Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</td><td>An integer indicating the Keyfactor Command reference ID for the metadata field. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the metadata field. This name appears in interfaces where you can use metadata, such as certificate details dialogs, alert dialogs, certificate imports and certificate requests. Once this field has a value associated with it for at least one certificate, you cannot change this name. The metadata name field cannot contain spaces; dashes and underscores are supported.</td></tr>
<tr><td>Description</td><td>A string indicating the description for the metadata field.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type of the metadata field. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>String</td></tr><tr><td>2</td><td>Integer</td></tr><tr><td>3</td><td>Date</td></tr><tr><td>4</td><td>Boolean</td></tr><tr><td>5</td><td>Multiple Choice</td></tr><tr><td>6</td><td>Big Text</td></tr></table></td></tr>
<tr><td>Hint</td><td>A string indicating a short hint for the metadata field. This hint appears in unpopulated metadata string, integer, big text and date fields on editing interfaces to provide the user with a clue as to what type of data should be entered in the field.
This field is only supported for metadata fields with data types string, integer, date or big text.</td></tr>
<tr><td>Validation</td><td>A string containing a regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:
<code>^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$</code>
This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>This field is only supported for metadata fields with data type string.

Tip: If a template specific option is set for a given metadata field, that takes precedence over the global options. The template-specific regular expression will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Enrollment</td><td>An integer indicating how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr>
<tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr>
<tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr>
</table>

Tip: If a template-specific handling is set for a given metadata field, it takes precedence over this global setting. The template-specific handling will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Message</td><td>A string containing a message to present when a user enters information in a metadata field that does not match the specified regular expression (Validation field).

Tip: If a template-specific regular expression message is set for a given metadata field, it takes precedence over this global regular expression message. The template-specific message will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).</td></tr>
<tr><td>Options</td><td>An array containing a comma separated list of values that should appear in the field dropdown for multiple choice fields.
This field is only supported for metadata fields with data type multiple choice.</td></tr>
</table>

Name	Description
	Tip: If a template-specific options are set for a given metadata field, these takes precedence over these global options. The template-specific options will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
DefaultValue	A string containing a default value with which to pre-populate the metadata field for new certificate requests made using PFX or CSR enrollment. Data type of Email will accept a comma separated list of email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, or *multiple choice*. **Tip:** If a template-specific default is set for a given metadata field, it takes precedence over this global default value. The template-specific default will be used in PFX and CSR enrollment requests using that template (see GET Templates ID on page 2377).
AllowAPI	This is considered deprecated and may be removed in a future release.
ExplicitUpdate	This is considered deprecated and may be removed in a future release.
DisplayOrder	An integer indicating the order in which the metadata field should be displayed on pages where the metadata fields are displayed (e.g. PFX enrollment, certificate details).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.23 Monitoring

The Monitoring component of the Keyfactor API provides a set of methods to support management of CRL and OCSP monitoring locations.

Table 465: Monitoring Endpoints

Endpoint	Method	Description	Link
/Revocation/	POST	Creates a new revocation monitoring	POST Monitoring

Endpoint	Method	Description	Link
		location.	Revocation on page 1822
/Revocation/	PUT	Edits the revocation monitoring location with the specified ID.	PUT Monitoring Revocation on page 1830
/Revocation/	GET	Returns details for all revocation monitoring location according to the provided filter and output parameters.	GET Monitoring Revocation on page 1817
/ResolveOSCP	POST	Resolves the given OCSP certificate authority.	POST Monitoring Resolve OSCP on page 1838
/Revocation/{id}	GET	Returns details for the revocation monitoring location with the specified ID.	GET Monitoring Revocation ID on the next page
/Revocation/{id}	DELETE	Deletes the revocation monitoring location with the specified ID.	DELETE Monitoring Revocation ID below
/Revocation/Test	POST	Tests the revocation monitoring alert with the specified ID.	POST Monitoring Revocation Test on page 1839
/Revocation/TestAll	POST	Tests the revocation monitoring alerts.	POST Monitoring Revocation Test All on page 1841

3.6.23.1 DELETE Monitoring Revocation ID

The DELETE Monitoring/Revocation/{id} method is used to delete the revocation monitoring location with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 466: DELETE Monitoring Revocation {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer that specifies the ID of the revocation monitoring location. Use the *GET /Monitoring/Revocation* method (see GET Monitoring Revocation on page 1817) to retrieve a list of all the revocation monitoring locations to determine the ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.23.2 GET Monitoring Revocation ID

The GET /Monitoring/Revocation/{id} method is used to retrieve the revocation monitoring location with the specified ID. This method returns HTTP 200 OK on a success with details of the location.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 467: GET Monitoring Revocation {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer that specifies the ID of the revocation monitoring location. Use the *GET /Monitoring/Revocation* method (see GET Monitoring Revocation on page 1817) to retrieve a list of all the revocation monitoring locations to determine the ID.

Table 468: GET Monitoring Revocation {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the revocation monitoring location.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the revocation monitoring location.</td></tr>
<tr><td>EndpointType</td><td>A string indicating the type of revocation monitoring endpoint: OCSP or CRL.</td></tr>
<tr><td>Location</td><td>A string indicating the location for the revocation monitoring endpoint.
For CRL endpoints, this can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location.
For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.</td></tr>
<tr><td>Email</td><td>An object indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>EnableReminder</td><td>A Boolean indicating whether to send email reminders for this location (true) or not (false).</td></tr><tr><td>WarningDays</td><td>An integer indicating the number of days before expiration to send the warning email.</td></tr><tr><td>Recipients</td><td>An array of strings indicating the email addresses to which the email reminders should be sent.</td></tr></table></td></tr>
<tr><td>Dashboard</td><td>An object indicating the configuration for display on the dashboard. Dashboard details are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Show</td><td>A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false).</td></tr><tr><td>WarningHours</td><td>An integer indicating the number of hours prior to expiration when the location begins to appear in a warning state on the dashboard.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>WarningHours is required if Show is set to true and EndpointType is CRL.
WarningHours is not supported for EndpointType OCSP.
If the Days or Weeks value is selected in the Management Portal, it will be converted to hours when stored in the database.</td></tr>
</table>
</td></tr>
<tr><td>Schedule</td><td>An object containing the inventory schedule set for the revocation monitoring location. Supported schedules are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>OCSPParameters</td><td>For OCSP endpoints only, an object indicating the OCSP endpoint configuration. OCSP endpoint details are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateAuthorityId</td><td>An integer indicating the Keyfactor Command reference ID of the CA in the database.
This value will be null on a response if the endpoint was configured using the FileName option.</td></tr>
<tr><td>AuthorityName</td><td>A string indicating the distinguished name of the CA. For example:
<code>CN=CorpIssuingCA1, DC=keyexample, DC=com</code></td></tr>
<tr><td>AuthorityNameId</td><td>A string indicating the base 64 encoded SHA1 hash of the AuthorityName.</td></tr>
<tr><td>AuthorityKeyId</td><td>A string indicating the base 64 encoded SHA1 hash of the CA certificate’s public key. This value is found in the CA’s certificate as the Subject Key</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>Identifier (SKID).</td></tr><tr><td>SampleSerialNumber</td><td>A string indicating the serial number of the certificate used to identity the CA.</td></tr><tr><td>FileName</td><td>A string indicating a file name for the certificate used to identify the CA for the OCSP endpoint if *CertificateContents* is used. This value will be null on a response if the endpoint was configured using the *CertificateAuthorityId* option.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.23.3 GET Monitoring Revocation

The GET /Monitoring/Revocation method is used to retrieve all revocation monitoring locations. This method returns HTTP 200 OK on a success with details of both OCSP and CRL revocation endpoint configurations.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 469: GET Monitoring Revocation Input Parameters

Name	In	Description

QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are:
		• DashboardWarningValue (WarningHours value)
		• DisplayName (Name)
		• EndpointType (1-CRL, 2-OCSP)
		• SendWarning (emailreminder) (true, false)
		• ShowOnDashboard (true, false)
		• Url
		• WarningDays
		
		Tip: To return all revocation monitoring locations of type CRL, use the following query:
		`EndpointType -eq 1`
		
		To return locations of type OCSP, use this query:
		`EndpointType -eq 2`

PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 470: GET Monitoring Revocation Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the revocation monitoring location.
Name	A string indicating the name of the revocation monitoring location.
EndpointType	A string indicating the type of revocation monitoring endpoint: OCSP or CRL.
Location	A string indicating the location for the revocation monitoring endpoint. For CRL endpoints, this can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location. For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.
Email	An object indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are: **Value** / **Description** EnableReminder — A Boolean indicating whether to send email reminders for this location (true) or not (false). WarningDays — An integer indicating the number of days before expiration to send the warning email. Recipients — An array of strings indicating the email addresses to which the email reminders should be sent.
Dashboard	An object indicating the configuration for display on the dashboard. Dashboard details are: **Value** / **Description** Show — A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false). WarningHours — An integer indicating the number of hours prior to expiration when the location begins to appear in a warning state on the dashboard.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>WarningHours is required if Show is set to true and EndpointType is CRL.
WarningHours is not supported for EndpointType OCSP.
If the Days or Weeks value is selected in the Management Portal, it will be converted to hours when stored in the database.</td></tr>
</table>
</td></tr>
<tr><td>Schedule</td><td>An object containing the inventory schedule set for the revocation monitoring location. Supported schedules are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>OCSPParameters</td><td>For OCSP endpoints only, an object indicating the OCSP endpoint configuration. OCSP endpoint details are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateAuthorityId</td><td>An integer indicating the Keyfactor Command reference ID of the CA in the database.
This value will be null on a response if the endpoint was configured using the FileName option.</td></tr>
<tr><td>AuthorityName</td><td>A string indicating the distinguished name of the CA. For example:
<code>CN=CorpIssuingCA1, DC=keyexample, DC=com</code></td></tr>
<tr><td>AuthorityNameId</td><td>A string indicating the base 64 encoded SHA1 hash of the AuthorityName.</td></tr>
<tr><td>AuthorityKeyId</td><td>A string indicating the base 64 encoded SHA1 hash of the CA certificate’s public key. This value is found in the CA’s certificate as the Subject Key</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>Identifier (SKID).</td></tr><tr><td>SampleSerialNumber</td><td>A string indicating the serial number of the certificate used to identity the CA.</td></tr><tr><td>FileName</td><td>A string indicating a file name for the certificate used to identify the CA for the OCSP endpoint if *CertificateContents* is used. This value will be null on a response if the endpoint was configured using the *CertificateAuthorityId* option.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.23.4 POST Monitoring Revocation

The POST /Monitoring/Revocation method is used to add a revocation monitoring location. This method returns HTTP 200 OK on a success with details of the location.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 471: POST Monitoring Revocation Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td>Path</td><td>Required. An integer indicating the Keyfactor Command reference ID of the revocation monitoring location.</td></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the revocation monitoring location.</td></tr>
<tr><td>EndpointType</td><td>Body</td><td>Required. A string indicating the type of revocation monitoring endpoint: OCSP or CRL.</td></tr>
<tr><td>Location</td><td>Body</td><td>Required. A string indicating the location for the revocation monitoring endpoint.
For CRL endpoints, this can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location.
Important: Because a "+" (plus sign) in a URL can represent either a space or a "+" Keyfactor Command has chosen to read "+" as a space. For CRL URLs that require a "+" (plus sign), rather than a space, replace plus signs in your CRL's URL with "<code>%2B</code>". Only replace the plus signs you don't wish to be treated as a space.
For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.</td></tr>
<tr><td>Email</td><td>Body</td><td>Required*. for CRL endpoints. An object indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>EnableReminder</td><td>A Boolean indicating whether to send email reminders for this location (true) or not (false). The default is false.</td></tr><tr><td>WarningDays</td><td>An integer indicating the number of days before expiration to send the warning email.</td></tr><tr><td>Recipients</td><td>An array of strings indicating the email addresses to which the email reminders should be sent.</td></tr></table></td></tr>
<tr><td>Dashboard</td><td>Body</td><td>Required. An object indicating the configuration for display on the dashboard. Dashboard details are:</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Show</td><td>Required. A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false). The default is false.</td></tr>
<tr><td>WarningHours</td><td>Required*. An integer indicating the number of hours prior to expiration when the location begins to appear in a warning state on the dashboard.
WarningHours is required if Show is set to true and EndpointType is CRL.
WarningHours is not supported for EndpointType OCSP.
If the Days or Weeks value is selected in the Management Portal, it will be converted to hours when stored in the database.</td></tr>
</table>
</td></tr>
<tr><td>Schedule</td><td>Body</td><td>An object containing the inventory schedule set for the revocation monitoring location. Supported schedules are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>OCSPPara-meters</td><td>Body</td><td>Required*. for OCSP endpoints. For OCSP endpoints only, an object indicating the OCSP endpoint configuration. OCSP endpoint details are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateContents</td><td>A string containing the base-64 encoded contents of a certificate issued by the CA.</td></tr>
<tr><td>CertificateAuthorityId</td><td>An integer indicating the Keyfactor Command reference ID of the CA in the database.
Use the GET /CertificateAuthority method (see <a>GET Certificate Authority on page 1201) to retrieve a list of all the CAs to determine the ID.</td></tr>
</table></td></tr>
</table>

Name	In	Description

Value	Description
AuthorityName	A string indicating the distinguished name of the CA. For example: `CN=CorpIssuingCA1, DC=keyexample, DC=com` Use the *POST /Monitoring/ResolveOCSP* method (see POST Monitoring Resolve OSCP on page 1838) with the *CertificateAuthorityId* or *CertificateContents* to resolve this value.
AuthorityNameId	A string indicating the base 64 encoded SHA1 hash of the *AuthorityName*. Use the *POST /Monitoring/ResolveOCSP* method (see POST Monitoring Resolve OSCP on page 1838) with the *CertificateAuthorityId* or *CertificateContents* to resolve this value.
AuthorityKeyId	A string indicating the base 64 encoded SHA1 hash of the CA certificate's public key. This value is found in the CA's certificate as the Subject Key Identifier (SKID). Use the *POST /Monitoring/ResolveOCSP* method (see POST Monitoring Resolve OSCP on page 1838) with the *CertificateAuthorityId* or *CertificateContents* to resolve this value.
SampleSerialNumber	A string indicating the serial number of the certificate used to identity the CA. Use the *POST /Monitoring/ResolveOCSP* method (see POST Monitoring Resolve OSCP on page 1838) with the *CertificateAuthorityId* or *CertificateContents* to resolve this value.
FileName	A string indicating a file name for the certificate used to identify the CA for the OCSP endpoint if *CertificateContents* is used.

Table 472: POST Monitoring Revocation Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string indicating the name of the revocation monitoring location.</td></tr>
<tr><td>EndpointType</td><td>A string indicating the type of revocation monitoring endpoint: OCSP or CRL.</td></tr>
<tr><td>Location</td><td>A string indicating the location for the revocation monitoring endpoint.
For CRL endpoints, this can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location.
For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.</td></tr>
<tr><td>Email</td><td>An object indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>EnableReminder</td><td>A Boolean indicating whether to send email reminders for this location (true) or not (false).</td></tr><tr><td>WarningDays</td><td>An integer indicating the number of days before expiration to send the warning email.</td></tr><tr><td>Recipients</td><td>An array of strings indicating the email addresses to which the email reminders should be sent.</td></tr></table></td></tr>
<tr><td>Dashboard</td><td>An object indicating the configuration for display on the dashboard. Dashboard details are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Show</td><td>A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false).</td></tr><tr><td>WarningHours</td><td>An integer indicating the number of hours prior to expiration when the location begins to appear in a warning state on the dashboard.
WarningHours is required if Show is set to true and EndpointType is CRL.
WarningHours is not supported for EndpointType OCSP.</td></tr></table></td></tr>
</table>

Name	Description
	Value / **Description**: If the *Days* or *Weeks* value is selected in the Management Portal, it will be converted to hours when stored in the database.
Schedule	An object containing the inventory schedule set for the revocation monitoring location. Supported schedules are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>OCSPParameters</td><td>For OCSP endpoints only, an object indicating the OCSP endpoint configuration. OCSP endpoint details are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateAuthorityId</td><td>An integer indicating the Keyfactor Command reference ID of the CA in the database.
This value will be null on a response if the endpoint was configured using the FileName option.</td></tr>
<tr><td>AuthorityName</td><td>A string indicating the distinguished name of the CA. For example:
<pre>CN=CorpIssuingCA1, DC=keyexample, DC=com</pre></td></tr>
<tr><td>AuthorityNameId</td><td>A string indicating the base 64 encoded SHA1 hash of the AuthorityName.</td></tr>
<tr><td>AuthorityKeyId</td><td>A string indicating the base 64 encoded SHA1 hash of the CA certificate’s public key. This value is found in the CA’s certificate as the Subject Key Identifier (SKID).</td></tr>
<tr><td>SampleSerialNumber</td><td>A string indicating the serial number of the certificate used to identity the CA.</td></tr>
<tr><td>FileName</td><td>A string indicating a file name for the certificate used to identify the CA for the OCSP endpoint if CertificateContents is used.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This value will be null on a response if the endpoint was configured using the *CertificateAuthorityId* option.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.23.5 PUT Monitoring Revocation

The PUT /Monitoring/Revocation method is used to modify the revocation monitoring location. This method returns HTTP 200 OK on a success with details of the location.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 473: PUT Monitoring Revocation {id} Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td>Path</td><td>Required. An integer indicating the Keyfactor Command reference ID of the revocation monitoring location.</td></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the revocation monitoring location.</td></tr>
<tr><td>EndpointType</td><td>Body</td><td>Required. A string indicating the type of revocation monitoring endpoint: OCSP or CRL.</td></tr>
<tr><td>Location</td><td>Body</td><td>Required. A string indicating the location for the revocation monitoring endpoint.
For CRL endpoints, this can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location.
Important: Because a "+" (plus sign) in a URL can represent either a space or a "+" Keyfactor Command has chosen to read "+" as a space. For CRL URLs that require a "+" (plus sign), rather than a space, replace plus signs in your CRL's URL with "<code>%2B</code>". Only replace the plus signs you don't wish to be treated as a space.
For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.</td></tr>
<tr><td>Email</td><td>Body</td><td>Required*. for CRL endpoints. An object indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>EnableReminder</td><td>A Boolean indicating whether to send email reminders for this location (true) or not (false). The default is false.</td></tr><tr><td>WarningDays</td><td>An integer indicating the number of days before expiration to send the warning email.</td></tr><tr><td>Recipients</td><td>An array of strings indicating the email addresses to which the email reminders should be sent.</td></tr></table></td></tr>
<tr><td>Dashboard</td><td>Body</td><td>Required. An object indicating the configuration for display on the dashboard. Dashboard details are:</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Show</td><td>Required. A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false). The default is false.</td></tr>
<tr><td>WarningHours</td><td>Required*. An integer indicating the number of hours prior to expiration when the location begins to appear in a warning state on the dashboard.
WarningHours is required if Show is set to true and EndpointType is CRL.
WarningHours is not supported for EndpointType OCSP.
If the Days or Weeks value is selected in the Management Portal, it will be converted to hours when stored in the database.</td></tr>
</table>
</td></tr>
<tr><td>Schedule</td><td>Body</td><td>An object containing the inventory schedule set for the revocation monitoring location. Supported schedules are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>OCSPParameters</td><td>Body</td><td>Required*. for OCSP endpoints. For OCSP endpoints only, an object indicating the OCSP endpoint configuration. OCSP endpoint details are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateContents</td><td>A string containing the base-64 encoded contents of a certificate issued by the CA.</td></tr>
<tr><td>CertificateAuthorityId</td><td>An integer indicating the Keyfactor Command reference ID of the CA in the database.
Use the GET /CertificateAuthority method (see GET Certificate Authority on page 1201) to retrieve a list of all the CAs to determine the ID.</td></tr>
</table></td></tr>
</table>

Name	In	Description

Value	Description
AuthorityName	A string indicating the distinguished name of the CA. For example: `CN=CorpIssuingCA1, DC=keyexample, DC=com` Use the *POST /Monitoring/ResolveOCSP* method (see POST Monitoring Resolve OSCP on page 1838) with the *CertificateAuthorityId* or *CertificateContents* to resolve this value.
AuthorityNameId	A string indicating the base 64 encoded SHA1 hash of the *AuthorityName*. Use the *POST /Monitoring/ResolveOCSP* method (see POST Monitoring Resolve OSCP on page 1838) with the *CertificateAuthorityId* or *CertificateContents* to resolve this value.
AuthorityKeyId	A string indicating the base 64 encoded SHA1 hash of the CA certificate's public key. This value is found in the CA's certificate as the Subject Key Identifier (SKID). Use the *POST /Monitoring/ResolveOCSP* method (see POST Monitoring Resolve OSCP on page 1838) with the *CertificateAuthorityId* or *CertificateContents* to resolve this value.
SampleSerialNumber	A string indicating the serial number of the certificate used to identity the CA. Use the *POST /Monitoring/ResolveOCSP* method (see POST Monitoring Resolve OSCP on page 1838) with the *CertificateAuthorityId* or *CertificateContents* to resolve this value.
FileName	A string indicating a file name for the certificate used to identify the CA for the OCSP endpoint if *CertificateContents* is used.

Table 474: PUT Monitoring Revocation {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string indicating the name of the revocation monitoring location.</td></tr>
<tr><td>EndpointType</td><td>A string indicating the type of revocation monitoring endpoint: OCSP or CRL.</td></tr>
<tr><td>Location</td><td>A string indicating the location for the revocation monitoring endpoint.
For CRL endpoints, this can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location.
For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.</td></tr>
<tr><td>Email</td><td>An object indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>EnableReminder</td><td>A Boolean indicating whether to send email reminders for this location (true) or not (false).</td></tr><tr><td>WarningDays</td><td>An integer indicating the number of days before expiration to send the warning email.</td></tr><tr><td>Recipients</td><td>An array of strings indicating the email addresses to which the email reminders should be sent.</td></tr></table></td></tr>
<tr><td>Dashboard</td><td>An object indicating the configuration for display on the dashboard. Dashboard details are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Show</td><td>A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false).</td></tr><tr><td>WarningHours</td><td>An integer indicating the number of hours prior to expiration when the location begins to appear in a warning state on the dashboard.
WarningHours is required if Show is set to true and EndpointType is CRL.
WarningHours is not supported for EndpointType OCSP.</td></tr></table></td></tr>
</table>

Name	Description
	Value / **Description**: If the *Days* or *Weeks* value is selected in the Management Portal, it will be converted to hours when stored in the database.
Schedule	An object containing the inventory schedule set for the revocation monitoring location. Supported schedules are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Name	Description
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>OCSPParameters</td><td>For OCSP endpoints only, an object indicating the OCSP endpoint configuration. OCSP endpoint details are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateAuthorityId</td><td>An integer indicating the Keyfactor Command reference ID of the CA in the database.
This value will be null on a response if the endpoint was configured using the FileName option.</td></tr>
<tr><td>AuthorityName</td><td>A string indicating the distinguished name of the CA. For example:
<pre>CN=CorpIssuingCA1, DC=keyexample, DC=com</pre></td></tr>
<tr><td>AuthorityNameId</td><td>A string indicating the base 64 encoded SHA1 hash of the AuthorityName.</td></tr>
<tr><td>AuthorityKeyId</td><td>A string indicating the base 64 encoded SHA1 hash of the CA certificate’s public key. This value is found in the CA’s certificate as the Subject Key Identifier (SKID).</td></tr>
<tr><td>SampleSerialNumber</td><td>A string indicating the serial number of the certificate used to identity the CA.</td></tr>
<tr><td>FileName</td><td>A string indicating a file name for the certificate used to identify the CA for the OCSP endpoint if CertificateContents is used.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>This value will be null on a response if the endpoint was configured using the *CertificateAuthorityId* option.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.23.6 POST Monitoring Resolve OSCP

The POST /Monitoring/ResolveOCSP method is used to resolve the given OCSP certificate authority. This method returns HTTP 200 OK on a success with details of the location.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 475: POST Monitoring Resolve OCSP Input Parameters

Name	In	Description
CertificateContents	Body	**Required***. A string indicating the certificate contents of a base-64 encoded PEM issued by the CA that you wish to resolve. One of either *CertificateContents* or *CertificateAuthorityId* is required, but not both.
CertificateAuthorityId	Body	**Required***. An integer indicating the Keyfactor Command reference ID of the CA in the database. Use the *GET /CertificateAuthority* method (see GET Certificate Authority on page 1201) to retrieve a list of all the CAs to determine the ID. One of either *CertificateContents* or *CertificateAuthorityId* is required, but not both.

Table 476: POST Monitoring Resolve OCSP Response Data

Name	Description
CertificateAuthorityId	An integer indicating the Keyfactor Command reference ID of the CA in the database.
AuthorityName	A string indicating the resolved certificate authority's name in X.500 format.
AuthorityNameId	A string indicating the hash of the certificate authority's name in hex format.
AuthorityKeyId	A string indicating the public key of the certificate authority's certificate.
SampleSerialNumber	A string indicating the serial number of the certificate authority's certificate.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.23.7 POST Monitoring Revocation Test

The POST /Monitoring/Revocation/Test method is used to test email alerts for a single configured revocation monitoring endpoint. This method returns HTTP 200 OK on a success with details about the email message generated for each alert.

Tip: Alerts are generated when a CRL is expired or in the warning period as defined by the number of days configured in the *Email Reminder* setting. For example, if you had a CRL that expired on June 30 and configured the email reminder period to 15 days before expiration, the warning status would begin for that CRL on June 15 and CRL alerts would be generated. A warning will also appear for any CRL or OCSP locations that produced an error or couldn't be resolved.
When alerts are tested or sent on a schedule, corresponding message are also written to the system event log on the server where the Keyfactor Command service runs. For testing, this is true regardless of the setting of the *SendAlerts* flag. Information is logged to the event log for both locations that are in a good state (e.g. CRL resolves and is not in a warning or expired state or response from OCSP) and locations that are in an error state (e.g. CRL resolves but is in the warning period or expired, CRL is expired, CRL or OCSP location does not resolve).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 477: POST Monitoring Revocation Test Input Parameters

Name	Description
AlertId	**Required**. An integer indicating the reference ID of revocation monitoring alert to test.
EvaluationDate	**Required**. A string indicating the evaluation date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). You can use the date to simulate running the alerts a month from now instead of today, for example, or put in a date far in the future to be sure you pick up some expiring CRLs for testing purposes.
SendAlerts	A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is *false*.

Table 478: POST Monitoring Revocation Test Response Data

Parameter	Description
RevocationMonitoringAlerts	An array of objects containing alert details resulting from the test. Revocation monitoring alert details are: **Name** / **Description** Subject — A string indicating the email message subject for each alert. The content of this subject is not user configurable. Message — A string indicating the email message that will be delivered for each alert. The content of this message is not user configurable. Recipients — An array of strings containing the recipient(s) for the alert.
AlertBuildResult	A string indicating the outcome of the test (e.g. Success).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.23.8 POST Monitoring Revocation Test All

The POST /Monitoring/Revocation/Test method is used to test email alerts for all configured revocation monitoring endpoints. Alerts are generated when a CRL is expired or in the warning period as defined by the number of days configured in the *Email Reminder* setting or when an OCSP endpoint is unreachable. For example, if you had a CRL that expired on June 30 and configured the email reminder period to 15 days before expiration, the warning status would begin for that CRL on June 15 and CRL alerts would be generated. This method returns HTTP 200 OK on a success with details about the email message generated for each alert.

Tip: Alerts are generated when a CRL is expired or in the warning period as defined by the number of days configured in the *Email Reminder* setting. For example, if you had a CRL that expired on June 30 and configured the email reminder period to 15 days before expiration, the warning status would begin for that CRL on June 15 and CRL alerts would be generated. A warning will also appear for any CRL or OCSP locations that produced an error or couldn't be resolved.
When alerts are tested or sent on a schedule, corresponding message are also written to the system event log on the server where the Keyfactor Command service runs. For testing, this is true regardless of the setting of the *SendAlerts* flag. Information is logged to the event log for both locations that are in a good state (e.g. CRL resolves and is not in a warning or expired state or response from OCSP) and locations that are in an error state (e.g. CRL resolves but is in the warning period or expired, CRL is expired, CRL or OCSP location does not resolve).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 479: POST Monitoring Revocation Test All Input Parameters

Name	Description
EvaluationDate	**Required**. A string indicating the evaluation date/time for the test. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). You can use the date to simulate running the alerts a month from now instead of today, for example, or put in a date far in the future to be sure you pick up some expiring CRLs for testing purposes.
SendAlerts	A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is *false*.

Table 480: POST Monitoring Revocation Test All Response Data

Parameter	Description
RevocationMonitoringAlerts	An array of objects containing alert details resulting from the test. Revocation monitoring alert details are: **Name** / **Description** Subject — A string indicating the email message subject for each alert. The content of this subject is not user configurable. Message — A string indicating the email message that will be delivered for each alert. The content of this message is not user configurable. Recipients — An array of strings containing the recipient(s) for the alert.
AlertBuildResult	A string indicating the outcome of the test (e.g. Success).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.24 Orchestrator Jobs

The Orchestrator Jobs component of the Keyfactor API includes methods necessary to schedule orchestrator jobs and view the results of jobs.

Table 481: Orchestrator Jobs Endpoints

Endpoint	Method	Description	Link
/JobStatus/Data	GET	Retrieves the results of a custom job using the provided information.	GET Orchestrator Jobs Job Status Data on the next page
/JobHistory	GET	Returns the details of history records on orchestrator jobs, including in-process jobs.	GET Orchestrator Jobs Job History on page 1844

Endpoint	Method	Description	Link
/ScheduledJobs	GET	Returns the details of active scheduled jobs, including in-process jobs.	GET Orchestrator Jobs Scheduled Jobs on page 1850
/Custom	POST	Schedules a custom job on the orchestrator using the provided information.	POST Orchestrator Jobs Custom on page 1854
/Reschedule	POST	Reschedules a failed orchestrator job.	POST Orchestrator Jobs Reschedule on page 1859
/Unschedule	POST	Unschedules an active orchestrator job.	POST Orchestrator Jobs Unschedule on page 1861
/Acknowledge	POST	Sets the status of a failed orchestrator job to acknowledged.	POST Orchestrator Jobs Acknowledge on page 1863
/Custom/Bulk	POST	Schedules a custom job on multiple orchestrator using the provided information.	POST Orchestrator Jobs Reschedule on page 1859

3.6.24.1 GET Orchestrator Jobs Job Status Data

The GET /OrchestratorJobs/JobStatus/Data method is used to return the data generated from a completed custom orchestrator (a.k.a. agent) job for a given job ID. This method returns HTTP 200 OK on a success with up to 2 MB of data from the job results.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Tip: This method is used to return the log results from a Fetch Logs job initiated for the Keyfactor Universal Orchestrator. When used to return results for a Fetch Logs job, the last 2 MB of data from the orchestrator's log file are returned as a string in the Data field.

Tip: If jobs for the Keyfactor Universal Orchestrator fail with messages similar to the following:

```
2021-08-05 10:47:23.1940
Keyfactor.Orchestrators.JobExecutors.OrchestratorJobExecutor [Debug] - Response
status code does not indicate success: 413 (Request Entity Too Large).

at System.Net.Http.HttpResponseMessage.EnsureSuccessStatusCode() in /_
/src/System.Net.Http/src/System/Net/Http/HttpResponseMessage.cs:line 172

at Keyfactor.Orchestrators.Services.HttpService.SendPostAsync[T](String uri,
Object requestData, Dictionary`2 headers) in F:\BuildAgents\Default1\_
work\24\s\src\OrchestratorServices\HttpService.cs:line 38
```

This indicates that the amount of data being returned on the job is greater than IIS on the Keyfactor Command server is configured to accept. You will need to make modifications to the IIS settings on your Keyfactor Command server to allow it to accept larger incoming pieces of content. See Fetch Logs on page 505 for more information.

Table 482: GET Orchestrator Jobs Job Status Data Input Parameters

Name	In	Description
jobHistoryId	Query	**Required**. The Keyfactor Command reference ID of the orchestrator job. Use the GET /OrchestratorJobs/JobHistory method (see GET Orchestrator Jobs Job History below) to retrieve a list of jobs to determine the job's history ID.

Table 483: GET Orchestrator Jobs Job Status Data Response Data

Name	Description
JobHistoryId	An integer indicate the Keyfactor Command reference ID used to track progress during orchestrator jobs.
Data	A string containing up to 2 MB of data returned from the custom job.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.24.2 GET Orchestrator Jobs Job History

The GET /OrchestratorJobs/JobHistory method is used to retrieve the status of an in progress or completed orchestrator (a.k.a. agent) job for a given job ID. This method returns HTTP 200 OK on a success with details of the requested orchestrator jobs.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 484: GET Orchestrator Jobs Job History Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Job History Search Feature on page 518. The query fields supported for this endpoint are: • AgentId (The GUID of the orchestrator. Run GET Agents on page 858 to find the ID.) • Agent (ClientMachine) • JobId • Result (Job result: 4-Failure, 3-Warning, 2-Success, 0-Unknown) • Status (Job status: 4-Acknowledged, 3-Completed, 2-InProcess, 1-Waiting, 0-Unknown, 5-CompletedWillRetry) • JobType (Management, Inventory, Discovery, SslDiscovery, Reenrollment, Monitoring, Sync, SSHSync) • Message • OperationStart (DateTime) • ScheduleType (Schedule: null (Immediately), I_(Interval), D_(Daily), W_(Weekly),M_(Monthly), O_(Once) • TargetPath
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *JobHistoryId*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 485: GET Orchestrator Jobs Job History Response Data

Name	Description
JobHistoryId	An integer indicating the Keyfactor Command reference ID used to track progress during orchestrator jobs.
AgentMachine	A string indicating the name of the server on which the agent or orchestrator is installed. This is not necessarily the actual DNS name of the server; the orchestrator may have been installed using an alternative as a reference name.
JobId	A string indicating the Keyfactor Command reference GUID assigned to the job.
Schedule	The inventory schedule for the most recently run instance of the orchestrator job. Possible values are:

Name	Description
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre><code>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</code></pre>
</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre><code>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</code></pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
</table>

Name	Description
JobType	A string indicating the job type (e.g. IISInventory).
Oper-ationStart	The time, in UTC, at which the orchestrator job started.
OperationEnd	The time, in UTC, at which the orchestrator job finished.
Message	A string providing the error message for the operation, if any.
Result	A string indicating the result of the orchestrator job. Possible values are: • Unknown • Success • Warning • Failure
Status	A string indicating the status of the orchestrator job. Possible values are: • Unknown • Waiting • In Process • Completed • Acknowledged • Completed Will Retry
StorePath	A string indicating the path to the certificate store on the target. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). See Adding or Modifying a Certificate Store on page 413 for more information.
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.24.3 GET Orchestrator Jobs Scheduled Jobs

The GET /OrchestratorJobs/ScheduledJobs method is used to retrieve orchestrator (a.k.a. agent) jobs that have active schedules. This includes jobs with ongoing schedules, such as inventory jobs that run periodically, and jobs that have been scheduled but have not yet been completed, such as management or discovery jobs. Both jobs that have not yet started and in-progress jobs are returned by this method. This method returns HTTP 200 OK on a success with details of the scheduled orchestrator jobs.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/read/

Table 486: GET Orchestrator Jobs Scheduled Jobs Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Job History Search Feature on page 518. The query fields supported for this endpoint are: • AgentId (The GUID of the orchestrator. Run GET Agents on page 858 to find the ID.) • Agent Machine (ClientMachine) • AgentPlatform (Platform types: 0-Unknown, 1-.NET, 2-Java, 3-Mac, 4-Android, 5-Native, 6-Bash, 7-Universal Orchestrator) • JobType (Management, Inventory, Discovery, SslDiscovery, Reenrollment, Monitoring, Sync, SSHSync) • AgentType (Use -contains comparison) (see capabilities in GET Agents on page 858) • Requested (DateTime) • ScheduleType (Schedule: null (Immediately), I_(Interval), D_(Daily), W_(Weekly),M_(Monthly), O_(Once) • TargetPath
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Requested*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 487: GET Orchestrator Jobs Scheduled Jobs Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID assigned to the job.
ClientMachine	A string containing the client machine name. The value for this will vary depending on the certificate store type. Typically, it is the hostname of the machine on which the store is located, but this may vary. See Adding or Modifying a Certificate Store on page 413 for more information.
Target	A string indicating the server name and path to the certificate store on the target (e.g. appsrvr162.keyexample.com - /opt/app/store.cer). The server name included in the *Target* is the value from the *ClientMachine*. The format for the path will vary depending on the certificate store type. For example, for a Java keystore, this will be a file path (e.g. /opt/myapp/store.jks), but for an F5 device, this will be a partition name on the device (e.g. Common). Some types of jobs (e.g. discovery) have no path. See Adding or Modifying a Certificate Store on page 413 for more information.

Name	Description
Schedule	

Name	Description
Requested	The time, in UTC, at which the orchestrator job was initiated and added to the job queue.
JobType	A string indicating the job type (e.g. IISInventory).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.24.4 POST Orchestrator Jobs Custom

The POST /OrchestratorJobs/Custom method is used to schedule a job with a custom job type on an orchestrator. This method returns HTTP 200 OK on a success with the GUID for the scheduled job.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/modify/

Tip: Data returned from a custom job once the job completes (e.g. a FetchLogs job) is stored in the Keyfactor Command database. To retrieve the data, use the *GET /OrchestratorJobs/JobHistory* method (see GET Orchestrator Jobs Job History on page 1844) to determine the *JobHistoryId* of the completed job and then use the *GET /OrchestratorJobs/JobStatus/Data* method (see GET Orchestrator Jobs Job Status Data on page 1843) to retrieve the data.

Table 488: POST Orchestrator Jobs Custom Input Parameters

Name	In	Description
AgentId	Body	**Required**. A string indicating the Keyfactor Command reference GUID of the orchestrator that will execute this job. To schedule a Fetch Logs job, use the *GET /Agents* method (see GET Agents on page 858) with a query of *Status -eq 2 and Capabilities -contains "LOGS"* to retrieve a list of your approved orchestrators with the LOGS capability to determine the ID of the orchestrator for which you want to retrieve logs. To schedule a job using your custom job type, use the *GET /Agents* method (see GET Agents on page 858) with a query of *Status -eq 2* to retrieve a list of your approved orchestrators to determine the ID of the orchestrator for which you want to schedule a custom job with your custom job type.
JobTypeName	Body	**Required**. A string indicating the reference name for the custom job type for the job. Use the *GET /JobTypes/Custom* method (see GET Custom Job Types on page 1598) to retrieve a list of your defined custom job types to determine the job type name to use.
Schedule	Body	An object containing the schedule for the custom job. The following schedule types are supported:

Name	Description
Off	Turn off a previously configured schedule.
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Weekly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint. The default is *Immediate*.

JobFields	Body	An object that set the values for any optional job fields configured for the custom job type. The *key* is the field name and the *value* is the value for the field.
		For example:
		
		`"JobFields": {`
		`  "Favorite Type of Pet": "Rat",`
		`  "Mother's Birthday": "1952-05-21"`
		`}`
		
		Note: If a job field has been configured with a default value and you wish to accept the default value, the field does not need to be submitted along with the POST /OrchestratorJobs/Custom request. The default value will be set automatically by Keyfactor Command. Submitting a value overrides the default value.
		
		Use the *GET /JobTypes/Custom* method (see GET Custom Job Types on page 1598) to retrieve a list of your defined custom job types to determine the job fields defined for the job type.
		
		Tip: The built-in Fetch Logs job does not have any optional job fields.

Table 489: POST Orchestrator Jobs Custom Response Data

Name	Description
JobId	A string indicating the Keyfactor Command reference GUID for the job.
OrchestratorId	A string indicating the Keyfactor Command reference GUID of the orchestrator that will execute this job.
JobTypeName	A string indicating the reference name for the custom job type for the job.
Schedule	An object containing the schedule for the custom job.
JobFields	An array of objects that set the values for any optional job fields configured for the custom job type.
RequestTimestmap	A string containing the date, in UTC, when the custom job was submitted.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.24.5 POST Orchestrator Jobs Reschedule

The POST /OrchestratorJobs/Reschedule method is used to reschedule a failed orchestrator job to retry. Jobs must have a result of Failed and a status of Completed or Acknowledged to be eligible for rescheduling. This endpoint returns 204 with no content upon success.

Only select types of jobs are eligible for rescheduling, including:

- Certificate Store Management
- Reenrollment
- Mac Auto-enrollment
- JKS, PEM and F5 Certificate Store Discovery
- SSH Synchronization
- Custom Jobs scheduled to run Weekly or Monthly

The following types of jobs cannot be rescheduled with this method:

- Certificate Store Inventory
  Change the inventory schedule on certificate stores using POST /CertificateStores/Schedule (see POST Certificate Stores Schedule on page 1462).
- Custom Jobs scheduled to run Immediately or Exactly Once
  A new custom job should be scheduled after the problem is resolved using POST

/OrchestratorJobs/Custom (see POST Orchestrator Jobs Custom on page 1854).

- Fetch Logs
  A new fetch logs job should be scheduled after the problem is resolved using POST /OrchestratorJobs/Custom (see POST Orchestrator Jobs Custom on page 1854).
- SSL Discovery and Monitoring
  Change the schedule on these using PUT /SSL/Networks (see PUT SSL Networks on page 2347).
- CA Synchronization for Remote CAs Managed with the Keyfactor Universal Orchestrator
  Change the schedule on these using PUT /CertificateAuthority (see PUT Certificate Authority on page 1248).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/modify/
/certificate_stores/schedule/
OR
/agents/management/modify/
/certificate_stores/schedule/#/ (where # is a reference to a specific certificate store container ID)
The required permissions will vary depending on the job type being rescheduled. The permissions shown above are appropriate for a certificate store management job.

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Tip: Be sure to resolve the problem that caused the job to fail before rescheduling it.

Table 490: POST Orchestrator Jobs Reschedule Input Parameters

Name	In	Description
JobAuditIds	Body	**Required***. An array of integers indicating the job IDs of the failed jobs that should be scheduled to retry. Use the *GET /OrchestratorJobs/JobHistory* method (see GET Orchestrator Jobs Job History on page 1844) with a query similar to the following to retrieve a list of all orchestrator jobs potentially eligible for rescheduling: `JobType -ne "Inventory" AND Result -eq "4" AND (Status -eq "4" OR Status -eq "3")` Either a list of one or more *JobAuditIds* or a *Query* is **required**, but not both.
Query	Body	**Required***. A string containing a query to identify the jobs to reschedule (e.g. field1 -eq value1 AND field2 -gt value2). Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, see to Scheduled Job Search Feature on page 512. Either a list of one or more *JobAuditIds* or a *Query* is **required**, but not both.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.24.6 POST Orchestrator Jobs Unschedule

The POST /OrchestratorJobs/Unschedule method is used to unschedule a scheduled orchestrator job. This endpoint returns 204 with no content upon success.

Only select types of jobs are eligible for unscheduling, including:

- Certificate Store Discovery and Management
- Reenrollment
- Mac Auto-enrollment
- Fetch Logs
- Custom Jobs

The following types of jobs cannot be unscheduled with this method:

- Certificate Store Inventory
  Change the inventory schedule on certificate stores using POST /CertificateStores/Schedule

(see POST Certificate Stores Schedule on page 1462).

- SSH Synchronization
  Change the schedule on these using PUT /SSH/ServerGroups (see PUT SSH Server Groups on page 2210).
- SSL Discovery and Monitoring
  Change the schedule on these using PUT /SSL/Networks (see PUT SSL Networks on page 2347).
- CA Synchronization for Remote CAs Managed with the Keyfactor Universal Orchestrator
  Change the schedule on these using PUT /CertificateAuthority (see PUT Certificate Authority on page 1248).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/modify/
/certificate_stores/schedule/
OR
/agents/management/modify/
/certificate_stores/schedule/#/ (where # is a reference to a specific certificate store container ID)

The required permissions will vary depending on the job type being unscheduled. The permissions shown above are appropriate for a certificate store management job.

Permissions for certificate stores can be set at either the global or certificate store container level. See Container Permissions on page 629 for more information about global vs container permissions.

Table 491: POST Orchestrator Jobs Unschedule Input Parameters

Name	In	Description
JobIds	Body	**Required***. An array of strings indicating the GUIDs for the job IDs of the jobs that should be unscheduled. Use the *GET /OrchestratorJobs/ScheduledJobs* method (see GET Orchestrator Jobs Scheduled Jobs on page 1850) with a query similar to the following to retrieve a list of all orchestrator jobs potentially eligible for unscheduling: `JobType -notcontains "SslDiscovery" AND JobType -notcontains "Monitoring" AND JobType -notcontains "Sync" AND JobType -notcontains "SSHSync" AND JobType -notcontains "Inventory"` Either a list of one or more *JobIds* or a *Query* is **required**, but not both.
Query	Body	**Required***. A string containing a query to identify the jobs to unschedule (e.g. field1 -eq value1 AND field2 -gt value2). Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, see to Scheduled Job Search Feature on page 512. Either a list of one or more *JobIds* or a *Query* is **required**, but not both.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.24.7 POST Orchestrator Jobs Acknowledge

The POST /OrchestratorJobs/Acknowledge method is used to set an orchestrator job to a status of acknowledged. Jobs must have a result of Failed or Warning and a status of Completed or CompletedWillRetry to be eligible for acknowledgment. Jobs that are in process or that have completed successfully cannot be set to a status of acknowledged. Setting a job to a status of acknowledged removes it from the count on the job history tab in the Keyfactor Command Management Portal (if the job falls within the count period defined by the *Job Failures and Warnings Age Out (days)* application setting—see Application Settings: Agents Tab on page 614). This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/modify/

Table 492: POST Orchestrator Jobs Acknowledge Input Parameters

Name	In	Description
JobAuditIds	Body	**Required***. An array of integers indicating the job IDs of the jobs that should be set to a status of acknowledged. Use the *GET /OrchestratorJobs/JobHistory* method (see GET Orchestrator Jobs Job History on page 1844) with a query similar to the following to retrieve a list of all orchestrator jobs potentially eligible for acknowledgment: `(Result -eq "4" OR Result -eq "3") AND (Status -eq "3" OR Status -eq "5")` Either a list of one or more *JobAuditIds* or a *Query* is **required**, but not both.
Query	Body	**Required***. A string containing a query to identify the jobs to acknowledge (e.g. field1 -eq value1 AND field2 -gt value2). Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, see to Scheduled Job Search Feature on page 512. Either a list of one or more *JobAuditIds* or a *Query* is **required**, but not both.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.24.8 POST Orchestrator Jobs Custom Bulk

The POST /OrchestratorJobs/Custom/Bulk method is used to schedule a job with a specified custom job type on multiple orchestrators at once. This method returns HTTP 200 OK on a success with the GUIDs for the scheduled jobs.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/agents/management/modify/

Tip: Data returned from a custom job once the job completes (e.g. a FetchLogs job) is stored in the Keyfactor Command database. To retrieve the data, use the *GET /OrchestratorJobs/JobHistory* method (see GET Orchestrator Jobs Job History on page 1844) to determine the *JobHistoryId* of the completed job and then use the *GET /Orches-*

tratorJobs/JobStatus/Data method (see GET Orchestrator Jobs Job Status Data on page 1843) to retrieve the data.

Table 493: POST Orchestrator Jobs Custom Bulk Input Parameters

Name	In	Description
Orches-tratorIds	Body	**Required**. A string indicating the Keyfactor Command referenced GUIDs of the orchestrators what will execute the jobs. To schedule a Fetch Logs job, use the *GET /Agents* method (see GET Agents on page 858) with a query of *Status -eq 2 and Capabilities -contains "LOGS"* to retrieve a list of your approved orchestrators with the LOGS capability to determine the ID of the orchestrators for which you want to retrieve logs. To schedule a job using your custom job type, use the *GET /Agents* method (see GET Agents on page 858) with a query of *Status -eq 2* to retrieve a list of your approved orchestrators to determine the ID of the orchestrators for which you want to schedule a custom job with your custom job type.
JobTypeNam-e	Body	**Required**. A string indicating the reference name for the custom job type for the job. A single bulk operation can only execute one job type. Use the *GET /JobTypes/Custom* method (see GET Custom Job Types on page 1598) to retrieve a list of your defined custom job types to determine the job type name to use.
Schedule	Body	An object containing the schedule for the custom job. The following schedule types are supported: **Name** / **Description** Off — Turn off a previously configured schedule. Immediate — A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*. Interval — A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job.</td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr></table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

The default is Immediate.
</td></tr>
<tr><td>JobFields</td><td>Body</td><td>An object that set the values for any optional job fields configured for the custom job type. The key is the field name and the value is the value for the field.
For example:
<pre>"JobFields": {
   "Favorite Type of Pet": "Rat",
   "Mother's Birthday": "1952-05-21"
}</pre>
Note: If a job field has been configured with a default value and you wish</td></tr>
</table>

Name	In	Description
		to accept the default value, the field does not need to be submitted along with the POST /OrchestratorJobs/Custom request. The default value will be set automatically by Keyfactor Command. Submitting a value overrides the default value. Use the *GET /JobTypes/Custom* method (see GET Custom Job Types on page 1598) to retrieve a list of your defined custom job types to determine the job fields defined for the job type. **Tip:** The built-in Fetch Logs job does not have any optional job fields.

Table 494: POST Orchestrator Jobs Custom Bulk Response Data

Name	Description
OrchestratorJobPairs	An array of objects containing identifying information for each orchestrator on which the job will be run. Orchestrator job pair parameters include: **Value** / **Description** JobId — A string indicating the Keyfactor Command reference GUID for the job. OrchestratorId — A string indicating the Keyfactor Command reference GUID of the orchestrator that will execute this job.
JobTypeName	A string indicating the reference name for the custom job type for the job.
Schedule	An object containing the schedule for the custom job.
JobFields	An array of objects indicating the values for any optional job fields configured for the custom job type.
RequestTimestmap	A string indicating the date, in UTC, when the custom job was submitted.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.25 PAM Providers

Privileged Access Management (PAM) functionality in the Keyfactor API allows for configuration of third party PAM providers to secure certificate stores and provide access credentials for certificate authorities, workflows, and other functions. PAM functionality is provided using custom PAM extensions. Keyfactor provides several PAM extensions on the publicly-facing Keyfactor GitHub:

https://keyfactor.github.io/integrations-catalog/content/pam

The PAM component of the Keyfactor API includes methods necessary to programmatically create, delete, edit, and list PAM providers and PAM provider types. PAM provider types must be created before PAM providers for them can be created.

Table 495: PamProviders Endpoints

Endpoint	Method	Description	Link
/	GET	Returns a list of all the configured PAM providers.	GET PAM Providers on page 1898
/	POST	Creates a new PAM provider.	POST PAM Providers on page 1916
/	PUT	Updates a PAM provider.	PUT PAM Providers on page 1935
/{id}	GET	Returns information for the specified PAM provider.	GET PAM Providers ID on the next page
/{id}	DELETE	Deletes a PAM provider.	DELETE PAM Providers ID below
/Types	GET	Returns a list of all available PAM provider types.	GET PAM Providers Types on page 1888
/Types	POST	Creates a new PAM provider type.	POST PAM Providers Types on page 1891
/Types	GET	Returns the PAM provider type with the specified ID.	GET PAM Providers Types ID on page 1954

3.6.25.1 DELETE PAM Providers ID

The DELETE /PamProviders/{id} method is used to delete a PAM provider by ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/pam/modify/
OR
/pam/modify/#/ (where # is a reference to a specific PAM provider ID)
Permissions for PAM providers can be set at either the global or PAM provider level. See PAM Permissions on page 631 for more information about global vs PAM provider permissions.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Table 496: DELETE PamProviders {id} v1 & v2 Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID for the PAM provider to be deleted. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the PAM provider's ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.25.2 GET PAM Providers ID

The GET /PamProviders/{id} method is used to return a PAM provider by ID. This method returns HTTP 200 OK on a success with details about the specified PAM provider.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/pam/read/
OR
/pam/read/#/ (where # is a reference to a specific PAM provider ID)
Permissions for PAM providers can be set at either the global or PAM provider level. See PAM Permissions on page 631 for more information about global vs PAM provider permissions.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the GET /PamProviders/{id} method has been redesigned to remove references to PAM associations with areas and containers.

Table 497: GET PamProviders {id} v2 Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the PAM provider to retrieve. Use the *GET /PAM/Providers* method (see GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the provider's ID.

Table 498: GET PamProviders {id} v2 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.
Name	A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.
ProviderType	An object containing details about the provider type for the provider. Provider type details include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string indicating the name of the provider type.
Provider-TypeParams	An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider records and records using PAM providers. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayNa-me	A string indicating the display name for the PAM provider type parameter. For provider types with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>authentication).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (True).  Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields: - Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.-com/SecretServer). - Secret Server Username: The name of the user that will be used to connect to SecretServer. - Secret Server Password: The password of the user that will be used to connect to SecretServer. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:</td></tr></table></td></tr></table>

Name	Description

Value	Description

Value	Description

```
{
   "Name": "Host",
   "DisplayName":"Secret
Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret
Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret
Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

```
{
   "Name":"SecretId",
```

Name	Description

Value	Description

Value	Description

```
   "DisplayName":"Secret
Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret
Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Value	Description
Provider-Type	An object containing details for the provider type. Provider type parameters include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the PAM provider type parameter.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM provider type.</td></tr>
<tr><td>ProviderTypeParams</td><td>An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new records.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. . Provider type parameter values include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr>
<tr><td>Value</td><td>A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or pass-</td></tr>
</table>
</td></tr>
</table>

Name	Description

Value	Description
	word resides).
InstanceId	An integer indicating the Keyfactor Command reference ID for the provider. If you are attaching to something with an integer Id, this will be used. This is considered deprecated and may be removed in a future release.
InstanceGuid	A string indicating the Keyfactor Command reference GUID for the provider. If you are attaching to something with a GUID ID, this will be used. This is considered deprecated and may be removed in a future release.
ProviderTypeParam	An object indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>PAM provider. For parameters with an InstanceLevel of True, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr>
<tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (True).
See example, above.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Remote</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
</table>

Version 1

Version 1 of the GET /PamProviders/{id} method includes the same capabilities as version 2 except it includes references to the deprecated parameters related to the area of Keyfactor Command to which the PAM provider applies.

Table 499: GET PamProviders {id} Iv1 nput Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the PAM provider to retrieve. Use the *GET /PAM/Providers* method (see GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the provider's ID.

Table 500: GET PamProviders {id} v1 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.
Name	A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.
Area	An integer indicating the area of Keyfactor Command the provider is used for. This is considered deprecated and may be removed in a future release.
ProviderType	An object containing details about the provider type for the provider. Provider type details include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string indicating the name of the provider type.
Provider-TypeParams	An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider records and records using PAM providers. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayNa-me	A string indicating the display name for the PAM provider type parameter. For provider types with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for authentication).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (True).  Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields: - Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.-com/SecretServer). - Secret Server Username: The name of the user that will be used to connect to SecretServer. - Secret Server Password: The password of the user that will be used to connect to SecretServer. </td></tr></table></td></tr></table>

Name	Description

Value	Description

Value	Description

Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:

```
{
   "Name": "Host",
   "DisplayName":"Secret
Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret
Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret
Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

Name	Description

Value	Description

Value	Description

```
{
   "Name":"SecretId",
   "DisplayName":"Secret
Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret
Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Provider-Type	An object containing details for the provider type. Provider type parameters include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the PAM provider type para-

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>meter.</td></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM provider type.</td></tr>
<tr><td>ProviderTypeParams</td><td>An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new records.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. . Provider type parameter values include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr>
<tr><td>Value</td><td>A string indicating the value set for the parameter</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>(e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr><tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the provider. If you are attaching to something with an integer Id, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the provider. If you are attaching to something with a GUID ID, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>ProviderTypeParam</td><td>An object indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of False, this name appears on the</td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of True, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr>
<tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (True).
See example, above.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Remote</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
<tr><td>SecureAreaId</td><td>An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any.
This is considered deprecated and may be removed in a future release.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.25.3 GET PAM Providers Types

The GET /PamProviders/Types method returns a list of all the PAM provider types that have been configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details about each PAM provider type. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/pam/read/
OR
/pam/read/#/ (where # is a reference to a specific PAM provider ID)
Permissions for PAM providers can be set at either the global or PAM provider level. See PAM Permissions on page 631 for more information about global vs PAM provider permissions.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852. The data returned in the response is the same for both versions.

Table 501: GET PamProviders Types v1 & v2 Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the PAM provider type.
Name	A string containing the name of the PAM provider type.
Parameters	An array of objects containing parameters set for the PAM provider type. Parameter details include:

Value	Description
Id	An integer indicating the ID of the parameter. Parameters will vary depending on your PAM extension. Built-in parameters include:

Value	Description
1	Private Ark Safe
2	PrivateArk Folder Name
3	PrivateArk Protected Password Name
4	Application ID
5	Secret Server Url
6	Rule Name
7	Thycotic Secret ID
8	Rule Key

Name	A string indicating the internal name for the PAM parameter.
DisplayName	A string indicating the display name for the PAM parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for authentication).
DataType	An integer indicating the data type for the parameter. Possible values are:

Name	Description

Value	Description
	• 1 = String • 2 = Secret
InstanceLevel	A Boolean that sets whether the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (true). **Example:** For Delinea when defining a PAM provider, you configure two Delinea-specific fields: • Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.com/SecretServer). • Secret Server Username: The name of the user that will be used to connect to SecretServer. • Secret Server Password: The password of the user that will be used to connect to SecretServer. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel-l=False like so: `{ "Name": "Host", "DisplayName":"Secret Server URL", "InstanceLevel":false, "DataType": "string" }, { "Name":"Username", "DisplayName":"Secret Server Username", "InstanceLevel":false, "DataType": "secret" }, { "Name":"Password", "DisplayName":"Secret Server Password", "InstanceLevel":false, "DataType": "secret" }`

Name	Description

Value	Description

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

```
{
   "Name":"SecretId",
   "DisplayName":"Secret Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.25.4 POST PAM Providers Types

The POST /PamProviders/Types method creates a new PAM provider type. This method returns HTTP 200 OK on a success with details about the PAM provider type.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/pam/modify/
OR
/pam/modify/#/ (where # is a reference to a specific PAM provider ID)
Permissions for PAM providers can be set at either the global or PAM provider level. See PAM Permissions on page 631 for more information about global vs PAM provider permissions.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852. The input parameters and response data returned are the same for both versions.

Table 502: POST PamProviders Types v1 & v2 Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string containing the name of the PAM provider type.</td></tr>
<tr><td>Parameters</td><td>Body</td><td>Required. An array of objects containing parameters set for the PAM provider type.
Parameter details include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Name</td><td>Required. A string indicating the internal name for the PAM parameter.</td></tr>
<tr><td>DisplayName</td><td>A string indicating the display name for the PAM parameter. For parameters with an InstanceLevel of False, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an InstanceLevel of True, this name appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for authentication).
If a DisplayName is not provided, the Name will be used as the DisplayName.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are:
1 = String
2 = Secret
The default is String.</td></tr>
<tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (true). The default is false.
Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields:
Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.com/SecretServer).</td></tr>
</table></td></tr>
</table>

Name	In	Description

Value	Description

- Secret Server Username: The name of the user that will be used to connect to SecretServer.
- Secret Server Password: The password of the user that will be used to connect to SecretServer.

Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:

```
{
   "Name": "Host",
   "DisplayName":"Secret Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

Name	In	Description
		Value / Description

```
{
   "Name":"SecretId",
   "DisplayName":"Secret Server Secret
ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Table 503: POST PamProviders Types v1 & v2 Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string containing the Keyfactor Command reference GUID for the PAM provider type.</td></tr>
<tr><td>Name</td><td>A string containing the name of the PAM provider type.</td></tr>
<tr><td>Parameters</td><td>An array of objects containing parameters set for the PAM provider type.
Parameter details include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the parameter. Parameters will vary depending on your PAM extension. Built-in parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>Private Ark Safe</td></tr>
<tr><td>2</td><td>PrivateArk Folder Name</td></tr>
<tr><td>3</td><td>PrivateArk Protected Password Name</td></tr>
<tr><td>4</td><td>Application ID</td></tr>
<tr><td>5</td><td>Secret Server Url</td></tr>
<tr><td>6</td><td>Rule Name</td></tr>
<tr><td>7</td><td>Thycotic Secret ID</td></tr>
<tr><td>8</td><td>Rule Key</td></tr>
</table></td></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM parameter.</td></tr>
<tr><td>DisplayName</td><td>A string indicating the display name for the PAM parameter. For parameters with an InstanceLevel of False, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an InstanceLevel of True, this name appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for authentication).</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are:</td></tr>
</table></td></tr>
</table>

Name	Description

Value	Description
	• 1 = String • 2 = Secret
InstanceLevel	A Boolean that sets whether the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (true). **Example:** For Delinea when defining a PAM provider, you configure two Delinea-specific fields: • Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.com/SecretServer). • Secret Server Username: The name of the user that will be used to connect to SecretServer. • Secret Server Password: The password of the user that will be used to connect to SecretServer. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel-l=False like so: `{ "Name": "Host", "DisplayName":"Secret Server URL", "InstanceLevel":false, "DataType": "string" }, { "Name":"Username", "DisplayName":"Secret Server Username", "InstanceLevel":false, "DataType": "secret" }, { "Name":"Password", "DisplayName":"Secret Server Password", "InstanceLevel":false, "DataType": "secret" }`

Name	Description

Value	Description

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

```
{
   "Name":"SecretId",
   "DisplayName":"Secret Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.25.5 GET PAM Providers

The GET /PamProviders method returns a list of all the PAM providers that have been configured in Keyfactor Command. This method returns HTTP 200 OK on a success with details about each PAM provider.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/pam/read/
OR
/pam/read/#/ (where # is a reference to a specific PAM provider ID)
Permissions for PAM providers can be set at either the global or PAM provider level. See PAM Permissions on page 631 for more information about global vs PAM provider permissions.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the GET /PamProviders method has been redesigned to remove references to PAM associations with areas and containers.

Table 504: GET PamProviders v2 Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • Area • Name • ProviderType • SecuredAreaId
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 505: GET PamProviders v2 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.
Name	A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.
ProviderType	An object containing details about the provider type for the provider. Provider type details include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string indicating the name of the provider type.
Provider-TypeParams	An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider records and records using PAM providers. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayNa-me	A string indicating the display name for the PAM provider type parameter. For provider types with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>authentication).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (True).  Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields: - Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.-com/SecretServer). - Secret Server Username: The name of the user that will be used to connect to SecretServer. - Secret Server Password: The password of the user that will be used to connect to SecretServer. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:</td></tr></table></td></tr></table>

Name	Description

Value	Description

Value	Description

```
{
   "Name": "Host",
   "DisplayName":"Secret
Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret
Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret
Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

```
{
   "Name":"SecretId",
```

Name	Description

Value	Description

Value	Description

```
   "DisplayName":"Secret
Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret
Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Value	Description
Provider-Type	An object containing details for the provider type. Provider type parameters include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the PAM provider type parameter.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM provider type.</td></tr>
<tr><td>ProviderTypeParams</td><td>An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new records.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. . Provider type parameter values include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr>
<tr><td>Value</td><td>A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or pass-</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>word resides).</td></tr><tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the provider. If you are attaching to something with an integer Id, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the provider. If you are attaching to something with a GUID ID, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>ProviderTypeParam</td><td>An object indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of False, this name appears on the PAM provider dialog for the parameter when a user creates a new</td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>PAM provider. For parameters with an InstanceLevel of True, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr>
<tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (True).
See example, above.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Remote</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
</table>

Version 1

Version 1 of the GET /PamProviders method includes the same capabilities as version 2 except it includes references to the deprecated parameters related to the area of Keyfactor Command to which the PAM provider applies.

Table 506: GET PamProviders v1 Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • Area • Name • ProviderType • SecuredAreaId
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 507: GET PamProviders v1 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.
Name	A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.
Area	An integer indicating the area of Keyfactor Command the provider is used for. This is considered deprecated and may be removed in a future release.
ProviderType	An object containing details about the provider type for the provider. Provider type details include: **Value** / **Description** Id — A string indicating the Keyfactor Command reference GUID for the provider type. Name — A string indicating the name of the provider type. ProviderTypeParams — An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider records and records using PAM providers. Provider type parameters values include: **Value** / **Description** Id — An integer indicating the Keyfactor Command reference ID for the PAM provider type. Name — A string indicating the internal name for the PAM provider type parameter. DisplayName — A string indicating the display name for the PAM provider type parameter. For provider types with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for authentication).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (True).  Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields: - Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.-com/SecretServer). - Secret Server Username: The name of the user that will be used to connect to SecretServer. - Secret Server Password: The password of the user that will be used to connect to SecretServer. </td></tr></table></td></tr></table>

Name	Description

Value	Description

Value	Description

Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:

```
{
   "Name": "Host",
   "DisplayName":"Secret
Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret
Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret
Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

Name	Description

Value	Description

Value	Description

```
{
   "Name":"SecretId",
   "DisplayName":"Secret
Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret
Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Value	Description
Provider-Type	An object containing details for the provider type. Provider type parameters include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the PAM provider type para-

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>meter.</td></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM provider type.</td></tr>
<tr><td>ProviderTypeParams</td><td>An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new records.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. . Provider type parameter values include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr>
<tr><td>Value</td><td>A string indicating the value set for the parameter</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>(e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr><tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the provider. If you are attaching to something with an integer Id, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the provider. If you are attaching to something with a GUID ID, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>ProviderTypeParam</td><td>An object indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of False, this name appears on the</td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of True, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr>
<tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (True).
See example, above.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Remote</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
<tr><td>SecureAreaId</td><td>An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any.
This is considered deprecated and may be removed in a future release.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.25.6 POST PAM Providers

The POST /PamProviders method creates a new PAM provider. This method returns HTTP 200 OK on a success with details for the new provider.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/pam/modify/
OR
/pam/modify/#/ (where # is a reference to a specific PAM provider ID)

Permissions for PAM providers and certificate stores can be set at either the global or PAM provider level. See PAM Permissions on page 631 for more information about global vs PAM provider permissions.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the POST /PamProviders method has been redesigned to remove references to PAM associations with areas and containers.

Table 508: POST PamProviders v2 Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.

⚠ Important: The name you give to your PAM provider in Keyfactor Command must match the name of the PAM provider as referenced in the manifest.json file (see <a>Installing Custom PAM Provider Extensions on page 743).</td></tr>
<tr><td>ProviderType</td><td>Body</td><td>Required. An object containing details about the provider type for the provider. Only the provider type ID is needed on input. Provider type details include:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>Required. A string indicating the Keyfactor Command reference GUID for the provider type.</td></tr></table></td></tr>
<tr><td>Provider-TypeParamValues</td><td>Body</td><td>Required*. An array of objects containing the values for the provider types specified by ProviderTypeParam. Values are only required in this field if the Remote parameter is set to false. Provider type parameter values include:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Value</td><td>Required. A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr><tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the provider. If you are attaching to something with an integer Id, this will be used.
This is considered deprecated and may be removed in a future release.</td></tr><tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the</td></tr></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>provider. If you are attaching to something with a GUID ID, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>ProviderTypeParam</td><td>An object indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>**Required**. An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM provider type parameter. For parameters with an</td></tr></table></td></tr></table>

Name	In	Description
		Value / Description Value / Description *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider. InstanceLevel: A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (True).

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>See example, above.</td></tr>
</table>
</td></tr>
</table>
Example: When creating a new PAM provider for Delinea local to Keyfactor Command, your POST body might look like:
<pre>
{
   "name": "PAMProviders.Delinea.PAMProvider",
   "providerType": {
      "id": "bd1762ce-3ea5-41fb-bfb4-1b6de6393fa3"
   },
   "providerTypeParamValues": [
      {
         "providerTypeParam": {
            "Id": 19
         },
         "Value": "https://MyDelineaURL"
      },
      {
         "providerTypeParam": {
            "Id": 20
         },
         "Value": "MyDelineaServiceAccountUser"
      },
      {
         "providerTypeParam": {
            "Id": 21
         },
         "Value": "MySu-
perSecretPasswordtoAccessDelinea"
      }
   ]
}
</pre>
</td></tr>
<tr><td>Remote</td><td>Body</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
</table>

Table 509: POST PamProviders v2 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.
Name	A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.
ProviderType	An object containing details about the provider type for the provider. Provider type details include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string indicating the name of the provider type.
Provider-TypeParams	An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider records and records using PAM providers. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayNa-me	A string indicating the display name for the PAM provider type parameter. For provider types with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>authentication).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (True).  Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields: - Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.-com/SecretServer). - Secret Server Username: The name of the user that will be used to connect to SecretServer. - Secret Server Password: The password of the user that will be used to connect to SecretServer. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:</td></tr></table></td></tr></table>

Name	Description

Value	Description

Value	Description

```
{
   "Name": "Host",
   "DisplayName":"Secret
Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret
Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret
Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

```
{
   "Name":"SecretId",
```

Name	Description

Value	Description

Value	Description

```
   "DisplayName":"Secret
Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret
Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Value	Description
Provider-Type	An object containing details for the provider type. Provider type parameters include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the PAM provider type parameter.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM provider type.</td></tr>
<tr><td>ProviderTypeParams</td><td>An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new records.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.</td></tr>
<tr><td>Remote</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
</table>

Version 1

Version 1 of the POST /PamProviders method includes the same capabilities as version 2 except it includes references to the deprecated parameters related to the area of Keyfactor Command to which the PAM provider applies.

Table 510: POST PamProviders v1 Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.

⚠ Important: The name you give to your PAM provider in Keyfactor Command must match the name of the PAM provider as referenced in the manifest.json file (see <a>Installing Custom PAM Provider Extensions on page 743).</td></tr>
<tr><td>ProviderType</td><td>Body</td><td>Required. An object containing details about the provider type for the provider. Only the provider type ID is needed on input. Provider type details include:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>Required. A string indicating the Keyfactor Command reference GUID for the provider type.</td></tr></table></td></tr>
<tr><td>Provider-TypeParamValues</td><td>Body</td><td>Required*. An array of objects containing the values for the provider types specified by ProviderTypeParam. Values are only required in this field if the Remote parameter is set to false. Provider type parameter values include:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Value</td><td>Required. A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr><tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the provider. If you are attaching to something with an integer Id, this will be used.
This is considered deprecated and may be removed in a future release.</td></tr><tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the</td></tr></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>provider. If you are attaching to something with a GUID ID, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>ProviderTypeParam</td><td>An object indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>**Required**. An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM provider type parameter. For parameters with an</td></tr></table></td></tr></table>

Name	In	Description
		Value / Description Value / Description *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider. InstanceLevel: A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (True).

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>See example, above.</td></tr>
</table>
</td></tr>
</table>
Example: When creating a new PAM provider for Delinea local to Keyfactor Command, your POST body might look like:
<pre>
{
   "name": "PAMProviders.Delinea.PAMProvider",
   "providerType": {
      "id": "bd1762ce-3ea5-41fb-bfb4-1b6de6393fa3"
   },
   "providerTypeParamValues": [
      {
         "providerTypeParam": {
            "Id": 19
         },
         "Value": "https://MyDelineaURL"
      },
      {
         "providerTypeParam": {
            "Id": 20
         },
         "Value": "MyDelineaServiceAccountUser"
      },
      {
         "providerTypeParam": {
            "Id": 21
         },
         "Value": "MySu-
perSecretPasswordtoAccessDelinea"
      }
   ]
}
</pre>
</td></tr>
<tr><td>Remote</td><td>Body</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
</table>

Table 511: POST PamProviders v2 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.
Name	A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.
Area	An integer indicating the area of Keyfactor Command the provider is used for. This is considered deprecated and may be removed in a future release.
ProviderType	An object containing details about the provider type for the provider. Provider type details include: **Value** / **Description** Id — A string indicating the Keyfactor Command reference GUID for the provider type. Name — A string indicating the name of the provider type. ProviderTypeParams — An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider records and records using PAM providers. Provider type parameters values include: **Value** / **Description** Id — An integer indicating the Keyfactor Command reference ID for the PAM provider type. Name — A string indicating the internal name for the PAM provider type parameter. DisplayName — A string indicating the display name for the PAM provider type parameter. For provider types with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for authentication).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (True).  Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields: - Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.-com/SecretServer). - Secret Server Username: The name of the user that will be used to connect to SecretServer. - Secret Server Password: The password of the user that will be used to connect to SecretServer. </td></tr></table></td></tr></table>

Name	Description

Value	Description

Value	Description

Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:

```
{
   "Name": "Host",
   "DisplayName":"Secret
Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret
Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret
Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

Name	Description

Value	Description

Value	Description

```
{
   "Name":"SecretId",
   "DisplayName":"Secret
Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret
Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Provider-Type

An object containing details for the provider type. Provider type parameters include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the PAM provider type para-

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>meter.</td></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM provider type.</td></tr>
<tr><td>ProviderTypeParams</td><td>An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new records.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.</td></tr>
<tr><td>Remote</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
<tr><td>SecureAreaId</td><td>An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any.
This is considered deprecated and may be removed in a future release.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.25.7 PUT PAM Providers

The PUT /PamProviders method updates an existing PAM provider. This method returns HTTP 200 OK on a success with details for the updated provider.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/pam/modify/
OR
/pam/modify/#/ (where # is a reference to a specific PAM provider ID)

Permissions for PAM providers and certificate stores can be set at either the global or PAM provider level. See PAM Permissions on page 631 for more information about global vs PAM provider permissions.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Version 2 of the PUT /PamProviders method has been redesigned to remove references to PAM associations with areas and containers.

Table 512: PUT PamProviders v2 Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td>Body</td><td>Required. An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.

⚠ Important: The name you give to your PAM provider in Keyfactor Command must match the name of the PAM provider as referenced in the manifest.json file (see <a>Installing Custom PAM Provider Extensions on page 743).</td></tr>
<tr><td>ProviderType</td><td>Body</td><td>Required. An object containing details about the provider type for the provider. Only the provider type ID is needed on input. Provider type details include:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>Required. A string indicating the Keyfactor Command reference GUID for the provider type.</td></tr>
</table></td></tr>
<tr><td>Provider-TypeParamValues</td><td>Body</td><td>Required*. An array of objects containing the values for the provider types specified by ProviderTypeParam. Values are only required in this field if the Remote parameter is set to false. Provider type parameter values include:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Value</td><td>Required. A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr>
<tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the provider. If you are attaching to something with an integer Id, this will be used.
This is considered deprecated and may</td></tr>
</table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>be removed in a future release.</td></tr><tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the provider. If you are attaching to something with a GUID ID, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>ProviderTypeParam</td><td>An object indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>Required. An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM</td></tr></table></td></tr></table>

Name	In	Description
		Value / Description Value / Description provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider. InstanceLevel: A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>certificate store to use the PAM provider (True).
See example, above.</td></tr>
</table>
</td></tr>
</table>
Example: When creating a new PAM provider for Delinea local to Keyfactor Command, your POST body might look like:
<pre>
{
   "name": "PAMProviders.Delinea.PAMProvider",
   "providerType": {
      "id": "bd1762ce-3ea5-41fb-bfb4-1b6de6393fa3"
   },
   "providerTypeParamValues": [
      {
         "providerTypeParam": {
            "Id": 19
         },
         "Value": "https://MyDelineaURL"
      },
      {
         "providerTypeParam": {
            "Id": 20
         },
         "Value": "MyDelineaServiceAccountUser"
      },
      {
         "providerTypeParam": {
            "Id": 21
         },
         "Value": "MySu-
perSecretPasswordtoAccessDelinea"
      }
   ]
}
</pre>
</td></tr>
<tr><td>Remote</td><td>Body</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
</table>

Table 513: PUT PamProviders v2 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.
Name	A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.
ProviderType	An object containing details about the provider type for the provider. Provider type details include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the provider type.
Name	A string indicating the name of the provider type.
Provider-TypeParams	An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider records and records using PAM providers. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayNa-me	A string indicating the display name for the PAM provider type parameter. For provider types with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>authentication).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (True).  Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields: - Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.-com/SecretServer). - Secret Server Username: The name of the user that will be used to connect to SecretServer. - Secret Server Password: The password of the user that will be used to connect to SecretServer. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:</td></tr></table></td></tr></table>

Name	Description

Value	Description

Value	Description

```
{
   "Name": "Host",
   "DisplayName":"Secret
Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret
Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret
Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

```
{
   "Name":"SecretId",
```

Name	Description

Value	Description

Value	Description

```
   "DisplayName":"Secret
Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret
Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Value	Description
Provider-Type	An object containing details for the provider type. Provider type parameters include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the PAM provider type parameter.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM provider type.</td></tr>
<tr><td>ProviderTypeParams</td><td>An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new records.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.</td></tr>
<tr><td>Remote</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
</table>

Version 1

Version 1 of the PUT /PamProviders method includes the same capabilities as version 2 except it includes references to the deprecated parameters related to the area of Keyfactor Command to which the PAM provider applies.

Table 514: PUT PamProviders v1 Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td>Body</td><td>Required. An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.

⚠ Important: The name you give to your PAM provider in Keyfactor Command must match the name of the PAM provider as referenced in the manifest.json file (see Installing Custom PAM Provider Extensions on page 743).</td></tr>
<tr><td>ProviderType</td><td>Body</td><td>Required. An object containing details about the provider type for the provider. Only the provider type ID is needed on input. Provider type details include:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>Required. A string indicating the Keyfactor Command reference GUID for the provider type.</td></tr></table></td></tr>
<tr><td>Provider-TypeParamValues</td><td>Body</td><td>Required*. An array of objects containing the values for the provider types specified by ProviderTypeParam. Values are only required in this field if the Remote parameter is set to false. Provider type parameter values include:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>Value</td><td>Required. A string indicating the value set for the parameter (e.g. the name of the CyberArk folder where the protected object that stores the username or password resides).</td></tr><tr><td>InstanceId</td><td>An integer indicating the Keyfactor Command reference ID for the provider. If you are attaching to something with an integer Id, this will be used.
This is considered deprecated and may</td></tr></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>be removed in a future release.</td></tr><tr><td>InstanceGuid</td><td>A string indicating the Keyfactor Command reference GUID for the provider. If you are attaching to something with a GUID ID, this will be used. This is considered deprecated and may be removed in a future release.</td></tr><tr><td>ProviderTypeParam</td><td>An object indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>Required. An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.</td></tr><tr><td>Name</td><td>A string indicating the internal name for the PAM provider type parameter.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name for the PAM</td></tr></table></td></tr></table>

Name	In	Description
		Value / Description Value / Description provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider. InstanceLevel: A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>certificate store to use the PAM provider (True).
See example, above.</td></tr>
</table>
</td></tr>
</table>

Example: When creating a new PAM provider for Delinea local to Keyfactor Command, your POST body might look like:
<pre>
{
   "name": "PAMProviders.Delinea.PAMProvider",
   "providerType": {
      "id": "bd1762ce-3ea5-41fb-bfb4-1b6de6393fa3"
   },
   "providerTypeParamValues": [
      {
         "providerTypeParam": {
            "Id": 19
         },
         "Value": "https://MyDelineaURL"
      },
      {
         "providerTypeParam": {
            "Id": 20
         },
         "Value": "MyDelineaServiceAccountUser"
      },
      {
         "providerTypeParam": {
            "Id": 21
         },
         "Value": "MySu-
perSecretPasswordtoAccessDelinea"
      }
   ]
}
</pre>
</td></tr>
<tr><td>Remote</td><td>Body</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
</table>

Table 515: PUT PamProviders v1 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider. This ID is automatically set by Keyfactor Command.
Name	A string indicating the name of the PAM provider. This name is used to identify the PAM provider throughout Keyfactor Command.
Area	An integer indicating the area of Keyfactor Command the provider is used for. This is considered deprecated and may be removed in a future release.
ProviderType	An object containing details about the provider type for the provider. Provider type details include: **Value** / **Description** Id — A string indicating the Keyfactor Command reference GUID for the provider type. Name — A string indicating the name of the provider type. ProviderTypeParams — An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider records and records using PAM providers. Provider type parameters values include: **Value** / **Description** Id — An integer indicating the Keyfactor Command reference ID for the PAM provider type. Name — A string indicating the internal name for the PAM provider type parameter. DisplayName — A string indicating the display name for the PAM provider type parameter. For provider types with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for authentication).</td></tr><tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are: 1 = String 2 = Secret </td></tr><tr><td>InstanceLevel</td><td>A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (True).  Example: For Delinea when defining a PAM provider, you configure two Delinea-specific fields: - Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.-com/SecretServer). - Secret Server Username: The name of the user that will be used to connect to SecretServer. - Secret Server Password: The password of the user that will be used to connect to SecretServer. </td></tr></table></td></tr></table>

Name	Description

Value	Description

Value	Description

Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:

```
{
   "Name": "Host",
   "DisplayName":"Secret
Server URL",
   "InstanceLevel":false,
   "DataType": "string"
},
{
   "Name":"Username",
   "DisplayName":"Secret
Server Username",
   "InstanceLevel":false,
   "DataType": "secret"
},
{
   "Name":"Password",
   "DisplayName":"Secret
Server Password",
   "InstanceLevel":false,
   "DataType": "secret"
}
```

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

Name	Description

Value	Description

Value	Description

```
{
   "Name":"SecretId",
   "DisplayName":"Secret
Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret
Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Provider-Type	An object containing details for the provider type. Provider type parameters include:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID for the PAM provider type para-

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>meter.</td></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM provider type.</td></tr>
<tr><td>ProviderTypeParams</td><td>An array of objects indicating parameters that the provider type uses for data input in Keyfactor Command when creating new records.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.</td></tr>
<tr><td>Remote</td><td>A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is false.</td></tr>
<tr><td>SecureAreaId</td><td>An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any.
This is considered deprecated and may be removed in a future release.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.25.8 GET PAM Providers Types ID

The GET /PamProviders/Types/{id} method returns the PAM provider type with the specified ID. This method returns HTTP 200 OK on a success with details about the specified PAM provider type. This method has only a v2 version.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/pam/read/
OR
/pam/read/#/ (where # is a reference to a specific PAM provider ID)
Permissions for PAM providers can be set at either the global or PAM provider level. See PAM Permissions on page 631 for more information about global vs PAM provider permissions.

Table 516: GET PamProviders Types {id} v2 Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID for the PAM provider to be deleted. Use the *GET /PamProviders* method (see GET PAM Providers on page 1898) to retrieve a list of all the PAM providers to determine the PAM provider's ID.

Table 517: GET PamProviders Types {id} v2 Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string containing the Keyfactor Command reference GUID for the PAM provider type.</td></tr>
<tr><td>Name</td><td>A string containing the name of the PAM provider type.</td></tr>
<tr><td>Parameters</td><td>An array of objects containing parameters set for the PAM provider type.
Parameter details include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the parameter. Parameters will vary depending on your PAM extension. Built-in parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>Private Ark Safe</td></tr>
<tr><td>2</td><td>PrivateArk Folder Name</td></tr>
<tr><td>3</td><td>PrivateArk Protected Password Name</td></tr>
<tr><td>4</td><td>Application ID</td></tr>
<tr><td>5</td><td>Secret Server Url</td></tr>
<tr><td>6</td><td>Rule Name</td></tr>
<tr><td>7</td><td>Thycotic Secret ID</td></tr>
<tr><td>8</td><td>Rule Key</td></tr>
</table></td></tr>
<tr><td>Name</td><td>A string indicating the internal name for the PAM parameter.</td></tr>
<tr><td>DisplayName</td><td>A string indicating the display name for the PAM parameter. For parameters with an InstanceLevel of False, this name appears on the PAM provider dialog when a user creates a new PAM provider. For parameters with an InstanceLevel of True, this name appears on the dialog when a user creates a new record using the PAM provider (e.g. a new certificate store using PAM for authentication).</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are:</td></tr>
</table></td></tr>
</table>

Name	Description

Value	Description
	• 1 = String • 2 = Secret
InstanceLevel	A Boolean that sets whether the parameter is used to define the underlying PAM provider (false) or a field that needs to be set to a value when configuring a record (e.g. a certificate store) to use the PAM provider (true). **Example:** For Delinea when defining a PAM provider, you configure two Delinea-specific fields: • Secret Server URL: The URL to the Secret Server vault instance, including port number if applicable (e.g. https://web-srvr38.keyexample.com/SecretServer). • Secret Server Username: The name of the user that will be used to connect to SecretServer. • Secret Server Password: The password of the user that will be used to connect to SecretServer. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel-l=False like so: `{ "Name": "Host", "DisplayName":"Secret Server URL", "InstanceLevel":false, "DataType": "string" }, { "Name":"Username", "DisplayName":"Secret Server Username", "InstanceLevel":false, "DataType": "secret" }, { "Name":"Password", "DisplayName":"Secret Server Password", "InstanceLevel":false, "DataType": "secret" }`

Name	Description

Value	Description

When you configure a certificate store to use Delinea as a credential provider, you enter the name of the secret field in Delinea referencing the protected object and you enter the ID of the projected object containing the username or password used to access the certificate store. Because these fields are configured on the certificate store level, they appear as InstanceLevel=True like so:

```
{
   "Name":"SecretId",
   "DisplayName":"Secret Server Secret ID",
   "InstanceLevel":true,
   "DataType": "string"
},
{
   "Name":"SecretFieldName",
   "DisplayName":"Secret Field Name",
   "InstanceLevel":true,
   "DataType": "string"
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in Delinea where the password is stored) are stored in the *ProviderTypeParamValues* array.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.26 Permissions

The Permissions component of the Keyfactor API includes the GET method to list all of the area permissions that are available to use to configure security settings for access to Keyfactor Command. This is mostly used for reference when configuring security settings.

Table 518: Security Roles Endpoints

Endpoint	Method	Description	Link
/	GET	Returns a list of all of the area permissions that are available to use to configure security settings for access to Keyfactor Command.	GET Permissions below

3.6.26.1 GET Permissions

The GET /Permissions method is used to list all area permissions available for use to control user access to all aspects of Keyfactor Command. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 519: GET Permissions Response Data

Name	Description

n/a	An array of strings listing the full permissions list in Keyfactor Command.
	For example:
	<pre>[
	"/",
	"/agents/",
	"/agents/auto_registration/",
	"/agents/auto_registration/modify/",
	"/agents/auto_registration/read/",
	"/agents/management/",
	"/agents/management/mac/",
	"/agents/management/mac/auto-enrollment/",
	"/agents/management/mac/auto-enrollment/management/",
	"/agents/management/mac/auto-enrollment/management/modify/",
	"/agents/management/mac/auto-enrollment/management/read/",
	...
	"/certificate_stores/",
	...
	"/ssl/read/",
	"/system_settings/",
	"/system_settings/modify/",
	"/system_settings/read/",
	"/workflows/",
	"/workflows/definitions/",
	]</pre>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.27 Permission Sets

The Permission Sets component of the Keyfactor API includes methods necessary to programmatically create, edit, retrieve, and delete permission sets within Keyfactor Command.

Permission sets are used to organize security roles (see Security Roles on page 2081) and provide compartmentalization on permissions. An unconfigured Keyfactor Command has one permission set—the *Global Permission Set*—that will contain all the security roles created in Keyfactor Command and allow users with administrative permissions to grant any of the possible permissions (see Security Role Permissions on page 632) in Keyfactor Command to other users. An implementation with two or more permission sets can be used to limit the actions that administrative users (or any users) can take.

Permission sets can only be managed with the Keyfactor API.

Example: An organization has a couple of administrators who should be allowed to make whatever changes are necessary to Keyfactor Command and a handful of other administrators who need full control to Keyfactor Command but should not be able to access certain sensitive features of the system. However, this second set of administrators do need to be able to change permissions of users on occasion. To meet this need, the organization decides to use permission sets:

- The *Global Permission Set* contains the following permission:

  ```
  /
  ```

  This is the base permission set that is created on installation. It allows any security roles created as part of the Global Permission Set to potentially contain any possible security permission within Keyfactor Command.
- The organization adds an *Operational Permission Set* which contains the following permissions:

  ```
  /agents/
  /application_settings/
  /certificate_stores/
  /certificates/
  /certificate_authorities/
  ```

```
/certificate_templates/
/dashboard/
/metadata/
/monitoring/
/portal/
/reports/
/security/
/scripts/
/ssl/
/workflows/
```

This allows any security roles created as part of the Operational Permission Set to potentially contain any of the permissions in the referenced areas of Keyfactor Command. Notice that security is among these areas. It does not allow security roles added to this permission set to be granted permissions in areas such as /auditing/ and /identity_providers/.

The organization creates these security roles:

- A Global Administrators security role in the Global Permission Set which grants full control to the system. This is created by default during the installation.
- An Operational Administrators security role in the Operational Permission Set which grants all the permissions in the Operational Permission Set.
- A Power Users security role in the Operational Permission Set which grants a large subset of the permissions in the Operational Permission Set, more granularly than grants to the administrators (e.g. /certificates/enrollment/csr/).
- A Viewers security role in the Operational Permission Set which grants a small subset of the permissions in the Operational Permission Set, more granularly than grants to the administrators (e.g. /certificates/collections/read/).

In this configuration, users who hold the Operational Administrators security role:

- Can edit the Power Users and Viewers security roles and change the permissions granted to those roles, but they cannot add any permissions that are not in the Operational Permission Set.
- Can edit the Operational Administrators security role, but can't add any permissions that aren't in the Operational Permission Set.
- Can add new claims for users, groups and other entities.
- Can add new security roles in the Operational Permission Set, referencing only permissions in that set.

- Can associate users, groups and other entities with the Power Users, Viewers and Operational Administrators security roles.
- Can remove role associations for users, groups and other entities for the Power Users, Viewers and Operational Administrators security roles.
- Cannot edit the Global Administrators role because it's not in the permission set to which their own security role belongs.
- Cannot add or edit permission sets.

Table 520: Permission Sets Endpoints

Endpoint	Method	Description	Link
/{id}	GET	Returns the permission set with the specified GUID.	GET Permission Sets ID below
/{id}	DELETE	Deletes the permission set with the specified GUID.	DELETE Permission Sets ID on the next page
/	GET	Returns a list of all the permission sets.	GET Permission Sets on page 1963
/	POST	Adds a new permission set into Keyfactor Command.	POST Permission Sets on page 1964
/	PUT	Updates a permission set.	PUT Permission Sets on page 1965

3.6.27.1 GET Permission Sets ID

The GET /PermissionSets/{id} method is used to return a permission set by GUID. This method returns HTTP 200 OK on a success with details for the specified permission set.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 521: GET Permission Sets{id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the permission set to retrieve. Use the *GET /PermissionSets* method (see GET Permission Sets on page 1963) to retrieve a list of all the permission sets to determine the permission set's ID.

Table 522: GET Permission Sets {id} Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the permission set.
Permissions	An array of strings containing the permissions assigned to the permission set. See Version Two Permission Model on page 632 for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.27.2 DELETE Permission Sets ID

The DELETE /PermissionSets/{id} method is used to delete the permission set with the specified GUID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 523: DELETE Permission Sets{id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the permission set to delete. Use the *GET /PermissionSets* method (see GET Permission Sets on the next page) to determine the ID of the permission set you wish to delete.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.27.3 GET Permission Sets

The GET /PermissionSets method is used to return a list of the permission sets defined in Keyfactor Command (see Permission Sets on page 1959). This method returns HTTP 200 OK on a success with details for the permission sets.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 524: GET Permission Sets Input Parameters

Name	In	Description
QueryString	Query	There are no query parsers for this endpoint..
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 525: GET Permission Sets Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the permission set.
Permissions	An array of strings containing the permissions assigned to the permission set. See Version Two Permission Model on page 632 for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.27.4 POST Permission Sets

The POST /PermissionSets method is used to create a new permission set in Keyfactor Command (see Permission Sets on page 1959). This method returns HTTP 200 OK on a success with the details of the new permission set.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 526: POST Permission Sets Input Parameters

Name	In	Description
Name	Body	**Required**. A string indicating the short name for the permission set.
Permissions	Body	**Required**. An array of strings containing the permissions assigned to the permission set. See Version Two Permission Model on page 632 for an overview of the possible permissions. For example: `"Permissions": [ "/agents/", "/certificate_stores/", "/certificates/", "/certificate_authorities/", "/certificate_templates/", "/dashboard/", "/metadata/", "/monitoring/", "/portal/" "/reports/", "/ssl/", "/workflows/", ]`

Table 527: POST Permission Sets Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the permission set.
Permissions	An array of strings containing the permissions assigned to the permission set. See Version Two Permission Model on page 632 for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.27.5 PUT Permission Sets

The PUT /PermissionSets method is used to update a permission set in Keyfactor Command (see Permission Sets on page 1959). This method returns HTTP 200 OK on a success with the details of the updated permission set.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 528: PUT Permission Sets Input Parameters

Name	Description
Id	**Required**. A string containing the Keyfactor Command reference GUID for the permission set.
Permissions	**Required**. An array of strings containing the permissions assigned to the permission set. See Version Two Permission Model on page 632 for an overview of the possible permissions. For example: `"Permissions": [ "/agents/", "/certificate_stores/", "/certificates/", "/certificate_authorities/", "/certificate_templates/", "/dashboard/", "/metadata/", "/monitoring/", "/portal/" "/reports/", "/ssl/", "/workflows/", ]`

Table 529: PUT Permission Sets Response Data

Name	Description
Id	A string containing the Keyfactor Command reference GUID for the permission set.
Permissions	An array of strings containing the permissions assigned to the permission set. See Version Two Permission Model on page 632 for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28 Reports

The Reports component of the Keyfactor API includes methods necessary to list, update, and schedule built-in reports as well as methods to create, update, list and delete custom reports.

Table 530: Reports Endpoints

Endpoint	Method	Description	Link
/{id}	GET	Returns the built-in report with the specified ID.	GET Reports ID on the next page
/Custom/{id}	DELETE	Deletes the custom report with the specified ID.	DELETE Reports Custom ID on page 1976
/Custom/{id}	GET	Returns the custom report with the specified ID.	GET Reports Custom ID on page 1977
/Schedules/{id}	DELETE	Deletes the schedule for the built-in report with the specified schedule ID.	DELETE Reports Schedules ID on page 1978
/Schedules/{id}	GET	Returns the schedule for the built-in report with the specified schedule ID.	GET Reports Schedules ID on page 1979
/{id}/Parameters	GET	Returns the parameters for the built-in report with the specified report ID.	GET Reports ID Parameters on page 1983
/{id}/Parameters	PUT	Updates the parameters for the built-in report with the specified report ID.	PUT Reports ID Parameters on page 1986
/	GET	Returns all built-in reports with filtering and output options.	GET Reports on page 1988
/	PUT	Updates the built-in report with the specified ID. Only some fields can be updated.	PUT Reports on page 1991
/Custom	GET	Returns all custom reports with filtering and output options.	GET Reports Custom on page 1994
/Custom	POST	Creates a custom report.	POST Reports Custom on page 1996
/Custom	PUT	Updates the custom report with the specified ID.	PUT Reports Custom on page 1998
/{id}/Schedules	GET	Returns the schedule for the built-in	GET Reports ID

Endpoint	Method	Description	Link
		report with the specified report ID.	Schedules on page 1999
/{id}/Schedules	POST	Creates a schedule for the built-in report with the specified report ID.	POST Reports ID Schedules on page 2004
/{id}/Schedules	PUT	Updates a schedule for the built-in report with the specified report ID.	PUT Reports ID Schedules on page 2014

3.6.28.1 GET Reports ID

The GET /Reports/{id} method is used to return the built-in report with the specified ID. This method returns HTTP 200 OK on a success with the details of the report.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/read/

Table 531: GET Reports {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer containing the Keyfactor Command reference ID for the report that should be retrieved. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID to use.

Table 532: GET Reports {id} Response Data

Name	Description
Id	An integer containing the Keyfactor Command reference ID for the report.
DisplayName	A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page, at the top of the page for the generated report, and on the menu. **Tip:** Exported reports use built-in names; modifying this value will not change the name that appears at the top of the exported version of a report (e.g. a PDF).
Description	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and at the top of the page for the generated report.
ReportPath	A string containing the name of the report as referenced when retrieving it via Logi Analytics.
VersionNumber	A string containing the version number for the report.
Categories	A string containing the report category or categories in which the report is found on the report manager page in the Keyfactor Command Management Portal. The possible values are: • CertificateCounts • CertificateLifecycle • CertificateLocations • PKIOperations • SecurityVulnerability • SSHKeys
ShortName	A string containing the short reference name for the report.
InNavigator	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false).
Favorite	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false).
RemoveDuplicates	A Boolean that indicates whether the report uses certificate de-duping logic in producing output (true) or not (false). **Tip:** When de-duplication is enabled for a report, the report results will include only the most recently issued certificate if there is more than one certi-

Name	Description
	ficate that matches the de-duplication criteria. De-duplication can only be enabled for reports that use certificate collections—the *UsesCollection* parameter. The *UsesCollection* parameter is not user-configurable. Certificate de-duping is configured on a certificate collection using the *DuplicationField* parameter (see POST Certificate Collections on page 1299). This corresponds to the Keyfactor Command Management Portal *Ignore renewed certificate results by* option on a certificate collection. Certificate collections may be configured to be de-duplicated based on the certificate common name, distinguished name, or principal name (or not at all). Only certificates that share all the EKUs (e.g. Client Authentication and Server Authentication) as well as the same CN, DN or UPN will be eliminated as duplicates. If a certificate has more than one EKU and at least one EKU does not match an otherwise similar certificate with matching CN, DN or UPN, it will not be eliminated.
UsesCollection	A Boolean that indicates whether the report uses a certificate collection as input for reporting (true) or not (false).
ReportParameter	An array of objects containing the parameters for the report. . Report parameters include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the report **parameter**.
ParameterName	A string containing the short reference name for the report parameter (e.g. EvalDate).
ParameterType	A string containing the type of the parameter. Possible values include: • Bool • CertAuth (certificate authorities) • Int • Metadata • OrchestratorPool • RelativeDate • SingleCA • SingleMetadata • SSHKeyType • Templates

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• TimePeriod</td></tr>
<tr><td>DisplayName</td><td>A string containing the display name for the parameter (e.g. Evaluation Date (UTC)).</td></tr>
<tr><td>Description</td><td>A string containing the description for the parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value for the parameter.

  Tip: Default values that are integers are also stored as strings in this parameter.</td></tr>
<tr><td>DisplayOrder</td><td>An integer indicating the order in which the parameters should be displayed on the scheduling page in Keyfactor Command, beginning with 0.</td></tr>
<tr><td>ParameterVisibility</td><td>A string indicating whether the parameter should be displayed in the Keyfactor Command Management Portal. The default value is Visible. The alternative setting is Hidden.</td></tr>
</table>
</td></tr>
<tr><td>Schedules</td><td>An array of objects containing the configured schedules for running the report, if any. Schedules include the following information:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the report schedule.</td></tr>
<tr><td>SendReport</td><td>A Boolean indicating whether the report will be sent to the email recipients configured in EmailRecipients (true) or not (false).</td></tr>
<tr><td>SaveReport</td><td>A Boolean indicating whether the report will be saved to the UNC path defined by SaveReportPath (true) or not (false).</td></tr>
<tr><td>SaveReportPath</td><td>A string containing the UNC path to which the report will be written, if configured.</td></tr>
<tr><td>ReportForm-</td><td>A string containing the report format selected for the scheduled</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description
at	report run. Supported values vary depending on the selected report and include: • PDF • Excel • CSV
KeyfactorSchedule	An object providing the schedule for the report. The schedule can be one of:

Name	Description
Off	Turn off a previously configured schedule.
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
Weekly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Name	Description
	Name: Description Name: Description Time: The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). Days: An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”). For example, every Monday, Wednesday and Friday at 5:30 pm: `"Weekly": { "Days": [ "Monday", "Wednesday", "Friday" ], "Time": "2023-11-27T17:30:00Z" }` Monthly: A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
 Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>EmailRe-cipients</td><td>An array of strings containing the email addresses of users configured as recipients of the scheduled report, if any.</td></tr>
<tr><td>RuntimePara-meters</td><td>An object containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>CertAuth</td><td>The certificate authority or authorities selected to report on.</td></tr><tr><td>EndDate</td><td>The end date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 0 days before today—meaning today).</td></tr><tr><td>EvalDate</td><td>The evaluation date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr><tr><td>Metadata</td><td>The custom metadata fields selected to include in the report.</td></tr><tr><td>MinCertCount</td><td>The minimum number of certificates that must have been issued for the given template before the template will be included in the report.</td></tr><tr><td>OrchestratorPool</td><td>The orchestrator pool selected to report on.</td></tr><tr><td>PeriodCount</td><td>The number of days, weeks or months selected to report on.</td></tr><tr><td>PeriodSize</td><td>The selected reporting period (day, weeks or months).</td></tr><tr><td>Requesters</td><td>The certificate requesters selected to include in the report.</td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SSHKeyType</td><td>The SSH key type(s) selected to report on.</td></tr>
<tr><td>StartDate</td><td>The start date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr>
<tr><td>TemplateIds</td><td>The Keyfactor Command identifiers for the templates to include in the report.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>AcceptedScheduleFormats</td><td>An array of strings containing the report formats supported for the report. Typically supported formats are PDF and Excel. Select reports support CSV format.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.2 DELETE Reports Custom ID

The DELETE /Reports/Custom/{id} method is used to delete the custom report link with the specified ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/modify/

Table 533: DELETE Reports Custom {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer containing the Keyfactor Command reference ID for the report link to be deleted. Use the *GET /Reports/Custom* method (see GET Reports Custom on page 1994) to retrieve a list of your custom report links to determine the report ID to use.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.3 GET Reports Custom ID

The GET /Reports/Custom/{id} method is used to return the custom report link with the specified ID. This method returns HTTP 200 OK on a success with the details of the report linkage.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/read/

Table 534: GET Reports Custom {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer containing the Keyfactor Command reference ID for the report link that should be retrieved. Use the *GET /Reports/Custom* method (see GET Reports Custom on page 1994) to retrieve a list of your custom reports to determine the report ID to use.

Table 535: GET Reports Custom {id} Response Data

Name	Description
CustomURL	A string containing the URL users should click from within Keyfactor Command to display the custom report (e.g. https://my-webserver.keyexample.com/mycustomreport/). **Tip:** Custom reports are automatically opened in a new browser tab.
Id	An integer containing the Keyfactor Command reference ID for the report link.
DisplayName	A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and on the menu.
Description	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page.
InNavigator	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false).
Favorite	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.4 DELETE Reports Schedules ID

The DELETE /Reports/Schedules/{id} method is used to delete the schedule for the built-in report with the specified schedule ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/modify/

Table 536: DELETE Reports Schedules {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the report schedule. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID and then *GET /Reports/{id}* (see GET Reports ID on page 1968) to retrieve the details for that report to determine the schedule ID to use.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.5 GET Reports Schedules ID

The GET /Reports/Schedules/{id} method is used to return the schedule for the built-in report with the specified **schedule** ID. This method returns HTTP 200 OK on a success with the details of the report schedule. Use the *GET /Reports/{id}/Schedules* method to return the schedule based on the **report** ID (see GET Reports ID Schedules on page 1999).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/read/

Table 537: GET Reports Schedules {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the report schedule. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID and then *GET /Reports/{id}* (see GET Reports ID on page 1968) to retrieve the details for that report to determine the schedule ID to use.

Table 538: GET Reports Schedules {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the report **schedule**.
SendReport	A Boolean indicating whether the report will be sent to the email recipients configured in *EmailRecipients* (true) or not (false).
SaveReport	A Boolean indicating whether the report will be saved to the UNC path defined by *SaveReportPath* (true) or not (false).
SaveReportPath	A string containing the UNC path to which the report will be written, if configured.
ReportFormat	A string containing the report format selected for the scheduled report run. Supported values vary depending on the selected report and include: • PDF • Excel • CSV
KeyfactorSchedule	An object providing the schedule for the report. The schedule can be one of:

Name	Description
Off	Turn off a previously configured schedule.
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Weekly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

Name	Description
	Name / **Description**

Name: (empty)

Description:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Days	An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).

For example, every Monday, Wednesday and Friday at 5:30 pm:

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

Monthly

A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Day	The number of the day, in the month, to run the job.

For example, on the first of every month at 5:30 pm:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>EmailRecipients</td><td>An array of strings containing the email addresses of users configured as recipients of the scheduled report, if any.</td></tr>
<tr><td>RuntimeParameters</td><td>An object containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CertAuth</td><td>The certificate authority or authorities selected to report on.</td></tr>
<tr><td>EndDate</td><td>The end date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 0 days before today—meaning today).</td></tr>
<tr><td>EvalDate</td><td>The evaluation date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr>
<tr><td>Metadata</td><td>The custom metadata fields selected to include in the report.</td></tr>
<tr><td>MinCertCount</td><td>The minimum number of certificates that must have been issued for the given template before the template will be included in the report.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>OrchestratorPool</td><td>The orchestrator pool selected to report on.</td></tr><tr><td>PeriodCount</td><td>The number of days, weeks or months selected to report on.</td></tr><tr><td>PeriodSize</td><td>The selected reporting period (day, weeks or months).</td></tr><tr><td>Requesters</td><td>The certificate requesters selected to include in the report.</td></tr><tr><td>SSHKeyType</td><td>The SSH key type(s) selected to report on.</td></tr><tr><td>StartDate</td><td>The start date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr><tr><td>TemplateIds</td><td>The Keyfactor Command identifiers for the templates to include in the report.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.6 GET Reports ID Parameters

The GET /Reports/{id}/Parameters method is used to return the parameters for the built-in report with the specified report ID. This method returns HTTP 200 OK on a success with the details of the report parameters.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/read/

Table 539: GET Reports {id} Parameters Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the built-in **report** the parameter is associated with. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID to use.

Table 540: GET Reports {id} Parameters Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the report **parameter**.
ParameterName	A string containing the short reference name for the report parameter (e.g. EvalDate).
ParameterType	A string containing the type of the parameter. Possible values include: • Bool • CertAuth (certificate authorities) • Int • Metadata • OrchestratorPool • RelativeDate • SingleCA • SingleMetadata • SSHKeyType • Templates • TimePeriod
DisplayName	A string containing the display name for the parameter (e.g. Evaluation Date (UTC)).
Description	A string containing the description for the parameter.
DefaultValue	A string containing the default value for the parameter. **Tip:** Default values that are integers are also stored as strings in this parameter.
DisplayOrder	An integer indicating the order in which the parameters should be displayed on the scheduling page in Keyfactor Command, beginning with 0.
ParameterVisibility	A string indicating whether the parameter should be displayed in the Keyfactor Command Management Portal. The default value is *Visible*. The alternative setting is *Hidden*.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.7 PUT Reports ID Parameters

The PUT /Reports/{id}/Parameters method is used to update the parameters for the built-in report with the specified report ID. Only some fields can be updated. This method returns HTTP 200 OK on a success with the details of the report parameters.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/modify/

Table 541: PUT Reports {id} Parameters Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the built-in **report** the parameter is associated with. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID to use.
Id	Body	**Required**. The Keyfactor Command reference ID of the report **parameter**. Use the *GET /Reports/{id}* (see GET Reports ID on page 1968) to retrieve the details for the desired report to determine the parameter ID to use.
DisplayName	Body	A string containing the display name for the parameter (e.g. Evaluation Date (UTC)).
Description	Body	A string containing the description for the parameter.
DefaultValue	Body	A string containing the default value for the parameter. **Tip:** Default values that are integers are also stored as strings in this parameter.

Table 542: PUT Reports {id} Parameters Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the report **parameter**.
ParameterName	A string containing the short reference name for the report parameter (e.g. EvalDate).
ParameterType	A string containing the type of the parameter. Possible values include: • Bool • CertAuth (certificate authorities) • Int • Metadata • OrchestratorPool • RelativeDate • SingleCA • SingleMetadata • SSHKeyType • Templates • TimePeriod
DisplayName	A string containing the display name for the parameter (e.g. Evaluation Date (UTC)).
Description	A string containing the description for the parameter.
DefaultValue	A string containing the default value for the parameter. **Tip:** Default values that are integers are also stored as strings in this parameter.
DisplayOrder	An integer indicating the order in which the parameters should be displayed on the scheduling page in Keyfactor Command, beginning with 0.
ParameterVisibility	A string indicating whether the parameter should be displayed in the Keyfactor Command Management Portal. The default value is *Visible*. The alternative setting is *Hidden*.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.8 GET Reports

The GET /Reports method is used to return all built-in reports with filtering and output options. This method returns HTTP 200 OK on a success with selected details of the reports. To view details of schedules and parameters for a report, use the *GET /Reports/{id}* method (see GET Reports ID on page 1968).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/read/

Table 543: GET Reports Input Parameters

Name	In	Description
AmmendedQuery	Query	This parameter is not available for use and will be deprecated in version 12.
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • Categories (CertificateCounts, CertificateLifecycle, Certificate Locations, PKIOperations, SecurityVulnerability,SSHKeys) • Favorite (true, false) • InNavigator (true, false) • Scheduled (Number of schedules) **Tip:** This method offers limited searchable fields. The most useful search is probably by category. For example, to return all the reports tagged with the PKI Operations category: `Categories -contains "PKIOperations"`
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 544: GET Reports Response Data

Name	Description
Id	An integer containing the Keyfactor Command reference ID for the report.
Scheduled	An integer indicating the number of schedules configured for the report.
DisplayName	A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page, at the top of the page for the generated report, and on the menu. **Tip:** Exported reports use built-in names; modifying this value will not change the name that appears at the top of the exported version of a report (e.g. a PDF).
Description	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and at the top of the page for the generated report.
ReportPath	A string containing the name of the report as referenced when retrieving it via Logi Analytics.
VersionNumber	A string containing the version number for the report.
Categories	A string containing the report category or categories in which the report is found on the report manager page in the Keyfactor Command Management Portal. The possible values are: • CertificateCounts • CertificateLifecycle • CertificateLocations • PKIOperations • SecurityVulnerability • SSHKeys
ShortName	A string containing the short reference name for the report.
InNavigator	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false).
Favorite	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false).
RemoveDuplicates	A Boolean that indicates whether the report uses certificate de-duping logic in producing output (true) or not (false).

Name	Description
	Tip: When de-duplication is enabled for a report, the report results will include only the most recently issued certificate if there is more than one certificate that matches the de-duplication criteria. De-duplication can only be enabled for reports that use certificate collections—the *UsesCollection* parameter. The *UsesCollection* parameter is not user-configurable. Certificate de-duping is configured on a certificate collection using the *DuplicationField* parameter (see POST Certificate Collections on page 1299). This corresponds to the Keyfactor Command Management Portal *Ignore renewed certificate results by* option on a certificate collection. Certificate collections may be configured to be de-duplicated based on the certificate common name, distinguished name, or principal name (or not at all). Only certificates that share all the EKUs (e.g. Client Authentication and Server Authentication) as well as the same CN, DN or UPN will be eliminated as duplicates. If a certificate has more than one EKU and at least one EKU does not match an otherwise similar certificate with matching CN, DN or UPN, it will not be eliminated.
UsesCollection	A Boolean that indicates whether the report uses a certificate collection as input for reporting (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.9 PUT Reports

The PUT /Reports method is used to update the built-in report with the specified report ID. Only some fields can be updated. To create or update a report schedule, use the *POST /Reports/{id}Schedules* (see POST Reports ID Schedules on page 2004) or *PUT /Reports/{id}Schedules* (see PUT Reports ID Schedules on page 2014) method. To update parameters for a built-in report, use the *PUT /Reports/{id}/Parameters* method (see PUT Reports ID Parameters on page 1986). This method returns HTTP 200 OK on a success with the details of the report.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/modify/

Table 545: PUT Reports Input Parameters

Name	In	Description
Id	Body	**Required**. The Keyfactor Command reference ID of the built-in report that should be updated. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID to use.
InNavigator	Body	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false).
Favorite	Body	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false).
RemoveDuplicates	Body	A Boolean that indicates whether the report uses certificate de-duping logic in producing output (true) or not (false). **Tip:** When de-duplication is enabled for a report, the report results will include only the most recently issued certificate if there is more than one certificate that matches the de-duplication criteria. De-duplication can only be enabled for reports that use certificate collections—the *UsesCollection* parameter. The *UsesCollection* parameter is not user-configurable. Certificate de-duping is configured on a certificate collection using the *DuplicationField* parameter (see POST Certificate Collections on page 1299). This corresponds to the Keyfactor Command Management Portal "Ignore renewed certificate results by" option on a certificate collection. Certificate collections may be configured to be de-duplicated based on the certificate common name, distinguished name, or principal name (or not at all). Only certificates that share all the EKUs (e.g. Client Authentication and Server Authentication) as well as the same CN, DN or UPN will be eliminated as duplicates. If a certificate has more than one EKU and at least one EKU does not match an otherwise similar certificate with matching CN, DN or UPN, it will not be eliminated.

Table 546: PUT Reports Response Data

Name	Description
Id	An integer containing the Keyfactor Command reference ID for the report.
DisplayName	A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page, at the top of the page for the generated report, and on the menu. **Tip:** Exported reports use built-in names; modifying this value will not change the name that appears at the top of the exported version of a report (e.g. a PDF).
Description	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and at the top of the page for the generated report.
ReportPath	A string containing the name of the report as referenced when retrieving it via Logi Analytics.
VersionNumber	A string containing the version number for the report.
Categories	A string containing the report category or categories in which the report is found on the report manager page in the Keyfactor Command Management Portal. The possible values are: • CertificateCounts • CertificateLifecycle • CertificateLocations • PKIOperations • SecurityVulnerability • SSHKeys
ShortName	A string containing the short reference name for the report.
InNavigator	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false).
Favorite	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false).
RemoveDuplicates	A Boolean that indicates whether the report uses certificate de-duping logic in producing output (true) or not (false). **Tip:** When de-duplication is enabled for a report, the report results will include only the most recently issued certificate if there is more than one

Name	Description
	certificate that matches the de-duplication criteria. De-duplication can only be enabled for reports that use certificate collections—the *UsesCollection* parameter. The *UsesCollection* parameter is not user-configurable. Certificate de-duping is configured on a certificate collection using the *DuplicationField* parameter (see POST Certificate Collections on page 1299). This corresponds to the Keyfactor Command Management Portal *Ignore renewed certificate results by* option on a certificate collection. Certificate collections may be configured to be de-duplicated based on the certificate common name, distinguished name, or principal name (or not at all). Only certificates that share all the EKUs (e.g. Client Authentication and Server Authentication) as well as the same CN, DN or UPN will be eliminated as duplicates. If a certificate has more than one EKU and at least one EKU does not match an otherwise similar certificate with matching CN, DN or UPN, it will not be eliminated.
UsesCollection	A Boolean that indicates whether the report uses a certificate collection as input for reporting (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.10 GET Reports Custom

The GET /Reports/Custom method is used to return all custom report links with filtering and output options. This method returns HTTP 200 OK on a success with the details of the report linkages.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/read/

Table 547: GET Reports Custom Input Parameters

Name	In	Description
AmmendedQuery	Query	This parameter is not available for use and will be deprecated in version 12.
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • Favorite (true, false) • InNavigator (true, false)
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 548: GET Reports Custom Response Data

Name	Description
CustomURL	A string containing the URL users should click from within Keyfactor Command to display the custom report (e.g. https://my-webserver.keyexample.com/mycustomreport/). **Tip:** Custom reports are automatically opened in a new browser tab.
Id	An integer containing the Keyfactor Command reference ID for the report link.
DisplayName	A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and on the menu.
Description	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page.
InNavigator	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false).
Favorite	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.11 POST Reports Custom

The POST /Reports/Custom method is used to add a link within Keyfactor Command to an externally hosted custom report. This method returns HTTP 200 OK on a success with the details of the report linkage.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/modify/

Table 549: POST Reports Custom Input Parameters

Name	In	Description
CustomURL	Body	**Required**. A string containing the URL users should click from within Keyfactor Command to display the custom report (e.g. https://my-webserver.keyexample.com/mycustomreport/). **Tip:** Custom reports are automatically opened in a new browser tab.
DisplayName	Body	**Required**. A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and on the menu.
Description	Body	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page.
InNavigator	Body	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false). The default is *false*.
Favorite	Body	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false). The default is *false*.

Table 550: POST Reports Custom Response Data

Name	Description
CustomURL	A string containing the URL users should click from within Keyfactor Command to display the custom report (e.g. https://my-webserver.keyexample.com/mycustomreport/). **Tip:** Custom reports are automatically opened in a new browser tab.
Id	An integer containing the Keyfactor Command reference ID for the report link.
DisplayName	A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and on the menu.
Description	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page.
InNavigator	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false).
Favorite	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.12 PUT Reports Custom

The PUT /Reports/Custom method is used to update the custom report link with the specified ID. This method returns HTTP 200 OK on a success with the details of the report linkage.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/modify/

Table 551: PUT Reports Custom Input Parameters

Name	In	Description
CustomURL	Body	**Required**. A string containing the URL users should click from within Keyfactor Command to display the custom report (e.g. https://my-webserver.keyexample.com/mycustomreport/). **Tip:** Custom reports are automatically opened in a new browser tab.
Id	Body	**Required**. An integer containing the Keyfactor Command reference ID for the report link. Use the *GET /Reports/Custom* method (see GET Reports Custom on page 1994) to retrieve a list of your custom report links to determine the report ID to use.
DisplayName	Body	**Required**. A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and on the menu.
Description	Body	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page.
InNavigator	Body	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false). The default is *false*.
Favorite	Body	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false). The default is *false*.

Table 552: PUT Reports Custom Response Data

Name	Description
CustomURL	A string containing the URL users should click from within Keyfactor Command to display the custom report (e.g. https://my-webserver.keyexample.com/mycustomreport/). **Tip:** Custom reports are automatically opened in a new browser tab.
Id	An integer containing the Keyfactor Command reference ID for the report link.
DisplayName	A string containing the display name for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page and on the menu.
Description	A string containing the description for the report. This appears in the Keyfactor Command Management Portal on the Report Manager page.
InNavigator	A Boolean that indicates whether the report has been configured to display on the Keyfactor Command Management Portal menu (true) or not (false).
Favorite	A Boolean that indicates whether the report has been marked as a favorite (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.13 GET Reports ID Schedules

The GET /Reports/{id}/Schedules method is used to return the schedule for the built-in report with the specified **report** ID. This method returns HTTP 200 OK on a success with the details of the report schedule. Use the *GET /Reports/Schedules/{id}* method to return the schedule based on the **schedule** ID (see GET Reports Schedules ID on page 1979).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/read/

Table 553: GET Reports {id} Schedules Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the built-in **report** the schedule is associated with. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID to use.

Table 554: GET Reports {id} Schedules Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the report **schedule**.
SendReport	A Boolean indicating whether the report will be sent to the email recipients configured in *EmailRecipients* (true) or not (false).
SaveReport	A Boolean indicating whether the report will be saved to the UNC path defined by *SaveReportPath* (true) or not (false).
SaveReportPath	A string containing the UNC path to which the report will be written, if configured.
ReportFormat	A string containing the report format selected for the scheduled report run. Supported values vary depending on the selected report and include: • PDF • Excel • CSV
KeyfactorSchedule	An object providing the schedule for the report. The schedule can be one of:

Name	Description
Off	Turn off a previously configured schedule.
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Weekly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre><code>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</code></pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>EmailRecipients</td><td>An array of strings containing the email addresses of users configured as recipients of the scheduled report, if any.</td></tr>
<tr><td>RuntimeParameters</td><td>An object containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CertAuth</td><td>The certificate authority or authorities selected to report on.</td></tr>
<tr><td>EndDate</td><td>The end date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 0 days before today—meaning today).</td></tr>
<tr><td>EvalDate</td><td>The evaluation date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr>
<tr><td>Metadata</td><td>The custom metadata fields selected to include in the report.</td></tr>
<tr><td>MinCertCount</td><td>The minimum number of certificates that must have been issued for the given template before the template will be included in the report.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>OrchestratorPool</td><td>The orchestrator pool selected to report on.</td></tr><tr><td>PeriodCount</td><td>The number of days, weeks or months selected to report on.</td></tr><tr><td>PeriodSize</td><td>The selected reporting period (day, weeks or months).</td></tr><tr><td>Requesters</td><td>The certificate requesters selected to include in the report.</td></tr><tr><td>SSHKeyType</td><td>The SSH key type(s) selected to report on.</td></tr><tr><td>StartDate</td><td>The start date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr><tr><td>TemplateIds</td><td>The Keyfactor Command identifiers for the templates to include in the report.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.14 POST Reports ID Schedules

The POST /Reports/{id}/Schedules method is used to create a schedule for the built-in report with the specified report ID. This method returns HTTP 200 OK on a success with the details of the report schedule.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/modify/

Table 555: POST Reports {id} Schedules Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the built-in **report** the schedule is associated with. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID to use.
SendReport	Body	A Boolean indicating whether the report will be sent to the email recipients configured in *EmailRecipients* (true) or not (false). The default is *false*.
SaveReport	Body	A Boolean indicating whether the report will be saved to the UNC path defined by *SaveReportPath* (true) or not (false). The default is *false*.
SaveReportPath	Body	**Required***. A string containing the UNC path to which the report will be written, if configured. **Note:** The path for saved reports must be provided in UNC format (\\servername\sharename\path) and must be accessible from the Keyfactor Command administration server. In addition: • Do not use a trailing "\" in the report path. • Ensure that the application pool service account has permission to write to the location where you want the outputted report to be saved. • When scheduling a report, schedule it for at least 10 minutes in advance of the current time if you wish it to run soon. If you want to run it faster than that, the Keyfactor Command Service will need to be restarted. This field is **required** if *SaveReport* is set to *true*.
ReportFormat	Body	**Required**. A string containing the report format selected for the scheduled report run. Supported values vary depending on the selected report and include: • PDF • Excel • CSV
KeyfactorSchedule	Body	**Required**. An object providing the schedule for the report. The schedule can be one of:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>

For example, every Monday, Wednesday and Friday at 5:30 pm:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
<tr><td>Monthl-
y</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>

For example, on the first of every month at 5:30 pm:


```
"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
</table>

  Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

For example:
</td></tr>
</table>

Name	In	Description

		<pre>"KeyfactorSchedule": {
		  "Monthly": {
		  "Day": 1,
		  "Time": "2021-07-01T17:00:00Z"
		  }
		},</pre>
		Or:
		<pre>"KeyfactorSchedule": {
		  "Weekly": {
		  "Days": [
		  "Monday",
		  "Thursday"
		  ],
		  "Time": "2021-07-01T17:00:00Z"
		  }
		},</pre>

EmailRecipients	Body	**Required*** . An array of strings containing the email addresses of users configured as recipients of the scheduled report, if any. For example:
		<pre>"EmailRecipients": [
		  "pkiadmins@keyexample.com",
		  "john.smith@keyexample.com"
		]</pre>
		This field is **required** if *SendReport* is set to *true*.

RuntimePara-meters	Body	**Required*** . An object containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as: Name: CertAuth — Description: The certificate authority or authorities selected to report on. Name: EndDate — Description: The end date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 0 days before

Name	In	Description

		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>today—meaning today).</td></tr><tr><td>EvalDate</td><td>The evaluation date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr><tr><td>Metadata</td><td>The custom metadata fields selected to include in the report.</td></tr><tr><td>MinCertCount</td><td>The minimum number of certificates that must have been issued for the given template before the template will be included in the report.</td></tr><tr><td>OrchestratorPool</td><td>The orchestrator pool selected to report on.</td></tr><tr><td>PeriodCount</td><td>The number of days, weeks or months selected to report on.</td></tr><tr><td>PeriodSize</td><td>The selected reporting period (day, weeks or months).</td></tr><tr><td>Requesters</td><td>The certificate requesters selected to include in the report.</td></tr><tr><td>SSHKeyType</td><td>The SSH key type(s) selected to report on.</td></tr><tr><td>StartDate</td><td>The start date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr><tr><td>TemplateIds</td><td>The Keyfactor Command identifiers for the templates to include in the report.</td></tr></table>
		For example:
		<pre>"RuntimeParameters": {
		  "StartDate": "60-Day-Before",
		  "EndDate": "7-Day-Before",
		  "Metadata": "AppOwnerFirstName, AppOwnerLastName",</pre>

Name	In	Description
		<pre> "Requesters": "jsmith" }</pre> This field is **required** for reports that have runtime parameters.

Table 556: POST Reports {id} Schedules Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the report **schedule**.
SendReport	A Boolean indicating whether the report will be sent to the email recipients configured in *EmailRecipients* (true) or not (false).
SaveReport	A Boolean indicating whether the report will be saved to the UNC path defined by *SaveReportPath* (true) or not (false).
SaveReportPath	A string containing the UNC path to which the report will be written, if configured.
ReportFormat	A string containing the report format selected for the scheduled report run. Supported values vary depending on the selected report and include: • PDF • Excel • CSV
KeyfactorSchedule	An object providing the schedule for the report. The schedule can be one of:

Name	Description
Off	Turn off a previously configured schedule.
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Weekly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

Name	Description
	Name / **Description**

Name: (empty)

Description:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Days	An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).

For example, every Monday, Wednesday and Friday at 5:30 pm:

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

Monthly

A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Day	The number of the day, in the month, to run the job.

For example, on the first of every month at 5:30 pm:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>EmailRecipients</td><td>An array of strings containing the email addresses of users configured as recipients of the scheduled report, if any.</td></tr>
<tr><td>RuntimeParameters</td><td>An object containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CertAuth</td><td>The certificate authority or authorities selected to report on.</td></tr>
<tr><td>EndDate</td><td>The end date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 0 days before today—meaning today).</td></tr>
<tr><td>EvalDate</td><td>The evaluation date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr>
<tr><td>Metadata</td><td>The custom metadata fields selected to include in the report.</td></tr>
<tr><td>MinCertCount</td><td>The minimum number of certificates that must have been issued for the given template before the template will be included in the report.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>OrchestratorPool</td><td>The orchestrator pool selected to report on.</td></tr><tr><td>PeriodCount</td><td>The number of days, weeks or months selected to report on.</td></tr><tr><td>PeriodSize</td><td>The selected reporting period (day, weeks or months).</td></tr><tr><td>Requesters</td><td>The certificate requesters selected to include in the report.</td></tr><tr><td>SSHKeyType</td><td>The SSH key type(s) selected to report on.</td></tr><tr><td>StartDate</td><td>The start date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr><tr><td>TemplateIds</td><td>The Keyfactor Command identifiers for the templates to include in the report.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.28.15 PUT Reports ID Schedules

The PUT /Reports/{id}/Schedules method is used to update the schedule for the built-in report with the specified report ID. This method returns HTTP 200 OK on a success with the details of the report schedule.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/reports/modify/

Table 557: PUT Reports {id} Schedules Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the built-in **report** the schedule is associated with. Use the *GET /Reports* method (see GET Reports on page 1988) to retrieve a list of your built-in reports to determine the report ID to use.
Id	Body	**Required**. An integer indicating the Keyfactor Command reference ID of the report **schedule**. Use the *GET /Reports/{id}* (see GET Reports ID on page 1968) to retrieve the details for the desired report to determine the schedule ID to use.
SendReport	Body	A Boolean indicating whether the report will be sent to the email recipients configured in *EmailRecipients* (true) or not (false). The default is *false*.
SaveReport	Body	A Boolean indicating whether the report will be saved to the UNC path defined by *SaveReportPath* (true) or not (false). The default is *false*.
SaveReportPath	Body	**Required***. A string containing the UNC path to which the report will be written, if configured. **Note:** The path for saved reports must be provided in UNC format (\\servername\sharename\path) and must be accessible from the Keyfactor Command administration server. In addition: • Do not use a trailing "\" in the report path. • Ensure that the application pool service account has permission to write to the location where you want the outputted report to be saved. • When scheduling a report, schedule it for at least 10 minutes in advance of the current time if you wish it to run soon. If you want to run it faster than that, the Keyfactor Command Service will need to be restarted. This field is **required** if *SaveReport* is set to *true*.
ReportFormat	Body	**Required**. A string containing the report format selected for the scheduled report run. Supported values vary depending on the selected report and include: • PDF • Excel • CSV
KeyfactorSchedule	Body	**Required**. An object providing the schedule for the report.

Name	In	Description

Name	Description
Off	Turn off a previously configured schedule.
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2023-11-25T23:30:00Z" }`
Weekly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters: Name / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). Days — An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”). For example, every Monday, Wednesday and Friday at 5:30 pm:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
 Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
For example:
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>

```
"KeyfactorSchedule": {
   "Monthly": {
      "Day": 1,
      "Time": "2021-07-01T17:00:00Z"
   }
},
```

Or:

```
"KeyfactorSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Thursday"
      ],
      "Time": "2021-07-01T17:00:00Z"
   }
},
```

</td></tr>
<tr><td>EmailRecipients</td><td>Body</td><td>Required*. An array of strings containing the email addresses of users configured as recipients of the scheduled report, if any. For example:

```
"EmailRecipients": [
   "pkiadmins@keyexample.com",
   "john.smith@keyexample.com"
]
```

This field is required if SendReport is set to true.</td></tr>
<tr><td>RuntimePara-meters</td><td>Body</td><td>Required*. An object containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CertAuth</td><td>The certificate authority or authorities selected to report on.</td></tr>
<tr><td>EndDate</td><td>The end date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 0 days before</td></tr>
</table>

</td></tr>
</table>

Name	In	Description

Name	Description
	today—meaning today).
EvalDate	The evaluation date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).
Metadata	The custom metadata fields selected to include in the report.
MinCertCount	The minimum number of certificates that must have been issued for the given template before the template will be included in the report.
OrchestratorPool	The orchestrator pool selected to report on.
PeriodCount	The number of days, weeks or months selected to report on.
PeriodSize	The selected reporting period (day, weeks or months).
Requesters	The certificate requesters selected to include in the report.
SSHKeyType	The SSH key type(s) selected to report on.
StartDate	The start date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).
TemplateIds	The Keyfactor Command identifiers for the templates to include in the report.

For example:

```
"RuntimeParameters": {
   "StartDate": "60-Day-Before",
   "EndDate": "7-Day-Before",
   "Metadata": "AppOwnerFirstName, AppOwnerLastName",
```

Name	In	Description
		<pre> "Requesters": "jsmith" }</pre> This field is **required** for reports that have runtime parameters.

Table 558: PUT Reports {id} Schedules Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the report **schedule**.
SendReport	A Boolean indicating whether the report will be sent to the email recipients configured in *EmailRecipients* (true) or not (false).
SaveReport	A Boolean indicating whether the report will be saved to the UNC path defined by *SaveReportPath* (true) or not (false).
SaveReportPath	A string containing the UNC path to which the report will be written, if configured.
ReportFormat	A string containing the report format selected for the scheduled report run. Supported values vary depending on the selected report and include: • PDF • Excel • CSV
KeyfactorSchedule	An object providing the schedule for the report. The schedule can be one of:

Name	Description
Off	Turn off a previously configured schedule.
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}
```

Weekly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre><code>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</code></pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>EmailRecipients</td><td>An array of strings containing the email addresses of users configured as recipients of the scheduled report, if any.</td></tr>
<tr><td>RuntimeParameters</td><td>An object containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CertAuth</td><td>The certificate authority or authorities selected to report on.</td></tr>
<tr><td>EndDate</td><td>The end date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 0 days before today—meaning today).</td></tr>
<tr><td>EvalDate</td><td>The evaluation date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr>
<tr><td>Metadata</td><td>The custom metadata fields selected to include in the report.</td></tr>
<tr><td>MinCertCount</td><td>The minimum number of certificates that must have been issued for the given template before the template will be included in the report.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>OrchestratorPool</td><td>The orchestrator pool selected to report on.</td></tr><tr><td>PeriodCount</td><td>The number of days, weeks or months selected to report on.</td></tr><tr><td>PeriodSize</td><td>The selected reporting period (day, weeks or months).</td></tr><tr><td>Requesters</td><td>The certificate requesters selected to include in the report.</td></tr><tr><td>SSHKeyType</td><td>The SSH key type(s) selected to report on.</td></tr><tr><td>StartDate</td><td>The start date selected for the reporting period to report on. This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr><tr><td>TemplateIds</td><td>The Keyfactor Command identifiers for the templates to include in the report.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.29 Scheduling

The Scheduling component of the Keyfactor API includes a method necessary to programmatically create and edit scheduled tasks in Keyfactor Command for task management that is not otherwise covered by an individual endpoint (e.g. PUT /Alerts/Expiration/Schedule).

Table 559: Scheduling Endpoints

Endpoint	Method	Description	Link
/	POST	Create or update task management schedules.	

3.6.29.1 POST Scheduling

The POST /Scheduling method is used to add or update the schedule for a task in the Keyfactor Command database. This method returns HTTP 200 OK on a success with details of the scheduled task.

This method is intended primarily to be used for updating CA health monitoring schedules. Although it is possible to update issued, expiration, key rotation, and pending alert schedules using this method, each of these has an endpoint dedicated to this purpose (see PUT Alerts Issued Schedule on page 973, PUT Alerts Expiration Schedule on page 934, PUT Alerts Key Rotation Schedule on page 1008, and PUT Alerts Pending Schedule on page 1042).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/modify/

Table 560: POST Scheduling Input Parameters

Name	In	Description
Id	Body	**Required***. An integer indicating the Keyfactor Command reference ID of the schedule to create or update. This value is **required** if you're updating an existing schedule. Use a GET method to determine this ID. For example, use the GET /CertificateAuthority/HealthMonitoring/Schedule method (see GET Certificate Authority Health Monitoring Schedule on page 1286) to retrieve the schedule for CA health monitoring to determine the health monitoring schedule ID.
ScheduleType	Body	**Required**. An integer indicating the type of schedule to be updated. Supported schedule types are: **Code** / **Category Name** 1 — Expiration Alert 2 — Pending Alert 10 — CA Health Monitoring Alert 20 — Issued Alert 22 — SSH Key Rotation Alert
Enabled	Body	A Boolean that indicates whether the schedule is enabled (true) or not (false). The default is *false*.
Interval	Body	**Required***. An integer indicating a job scheduled to run every x minutes with x equal to the specified value. One of either *Interval* or *TimeOfDay* is **required**.
TimeOfDay	Body	**Required***. A string indicating a job scheduled to run daily at the specified time of day. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). One of either *Interval* or *TimeOfDay* is **required**.

Table 561: POST Scheduling Response Data

Name	Description
id	An integer indicating the Keyfactor Command reference ID of the schedule.
Schedule	A string indicating the schedule set for the item. For an interval schedule, this will look like I_mm where mm is the number of minutes (e.g. I_30 for every 30 minutes). For daily schedules, this will look like D_hh:mm where hh:mm is the time to run the job (e.g. D_14:30 for daily at 2:30 pm).
ScheduleType	An integer indicating the type of schedule. Supported schedule types are: **Code** / **Category Name** 1 — Expiration Alert 2 — Pending Alert 10 — CA Health Monitoring Alert 20 — Issued Alert 22 — SSH Key Rotation Alert
Enabled	A Boolean that indicates whether the schedule is enabled (true) or not (false).
Name	A string indicating the type of job.
EntityId	This is considered deprecated and may be removed in a future release.
LastRun	This is considered deprecated and may be removed in a future release.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30 Security

The Security component of the Keyfactor API includes methods necessary to list, add, and delete security identities and their permissions which are used to control access to aspects of Keyfactor Command.

Table 562: Security Endpoints

Endpoint	Method	Description	Link
/Identities	POST	Adds a new security identity into Keyfactor Command.	POST Security Identities on page 2038
/Identities	GET	Returns all security identities with with sorting and filter options.	GET Security Identities on page 2034
/Identities/Lookup	GET	Validates that the identity with the specified name exists.	GET Security Identities Lookup on page 2033
/Identities/{id}	DELETE	Deletes the security identity with the specified ID.	DELETE Security Identities ID below
/Identities/{id}	GET	Returns permission details for the security identity with the specified ID.	GET Security Identities ID on the next page
/Containers/{id}/Roles	GET	Returns permission details for the certificate store container with the specified ID.	GET Security Containers ID Roles on page 2039
/Containers/{id}/Roles	POST	Sets the permissions of the certificate store container with the specified ID.	POST Security Containers ID Roles on page 2040
/Audit/Collections/{id}	GET	Returns permission details for the certificate collection with the specified ID.	GET Security Audit Collections ID on page 2042
/My	GET	Returns permission details for the current user, including certificate collections and certificate store containers.	GET Security My on page 2045

3.6.30.1 DELETE Security Identities ID

The DELETE /Security/Identities/{id} method is used to delete the security identity with the specified ID from Keyfactor Command. Use the *GET /Security/Identities* method (see GET Security Identities on page 2034) to determine the ID of the security identity you wish to delete. The current user's identity may not be deleted. This endpoint returns 204 with no content upon success.

Note: This endpoint is for managing legacy formatted Active Directory identities only and is retained for backwards compatibility. New applications should use the *Security Claims* set of endpoints for both Active Directory and other identity providers (see Security on page 2027).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

Table 563: DELETE Security Identities {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer containing the Keyfactor Command reference ID of the security identity that should be deleted from Keyfactor Command. Use the *GET /Security/Identity* method (see GET Security Identities on page 2034) to retrieve a list of all the security identities to determine the identity's ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30.2 GET Security Identities ID

The GET /Security/Identities/{id} method is used to return the security identities configured in Keyfactor Command with the specified ID. This method returns HTTP 200 OK on a success with the details of the security identity's permissions.

Note: This endpoint is for managing legacy formatted Active Directory identities only and is retained for backwards compatibility. New applications should use the *Security Claims* set of endpoints for both Active Directory and other identity providers (see Security on page 2027).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 564: GET Security Identities {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security identity to retrieve. Use the *GET /Security/Identities* method (see GET Security Identities on page 2034) to retrieve a list of all the security identities to determine the identity's ID.

Table 565: GET Security Identities {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Identity</td><td>A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example:
<code>KEYEXAMPLE\\PKI Administrators</code></td></tr>
<tr><td>SecuredAreaPermissions</td><td>An array of objects containing information about the global permissions granted to the security identity. Global permission information includes:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Permission</td><td>A string indicating the permission granted. In the case of global permissions, this is the name of the role followed by the level of permission granted, the choices for which vary depending on the role.</td></tr><tr><td>GrantedByRoles</td><td>An object containing a list of roles that grant that permission.</td></tr></table>For example:
<pre>"SecuredAreaPermissions": [
   {
      "Permission": "AdminPortal:Read",
      "GrantedByRoles": [
         "Read Only",
         "Staff"
      ]
   },
   {
      "Permission": "Reports:Read",
      "GrantedByRoles": [
         "Read Only"
      ]
   },
]</pre>For more information about global permissions, see <a>Security Roles and Claims on page 622.</td></tr>
<tr><td>CollectionPermissions</td><td>An array of objects containing information about the certificate collection permissions granted to the security identity. Collection permission information includes:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Permission</td><td>A string indicating the permission granted. In the case of collection permissions, this is the name of the certificate collection followed by the level of permission granted.</td></tr>
<tr><td>GrantedByRoles</td><td>An array containing a list of roles that grant that permission.</td></tr>
</table>
For example:
<pre>
"CollectionPermissions": [
   {
      "Permission": "Issued in the Last Week:Certificates_Read",
      "GrantedByRoles": [
         "Staff",
         "Power Users"
      ]
   },
   {
      "Permission": "Web Server Certs:Certificates_EditMetadata",
         "GrantedByRoles": [
         "Power Users"
      ]
   },
]
</pre>
For more information about collection permissions, see <a>Certificate Collection Permissions on page 627.
</td></tr>
<tr><td>ContainerPermissions</td><td>An array of objects containing information about the certificate store container permissions granted to the security identity. Container permission information includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Permission</td><td>A string indicating the permission granted. In the case of container permissions, this is the name of the certificate store container followed by the level of permission granted (read, schedule or modify).</td></tr>
<tr><td>GrantedByRoles</td><td>An array containing a list of roles that grant that permission.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	For example: `"ContainerPermissions": [ { "Permission": "IIS Personal:CertificateStoreManagement_Read", "GrantedByRoles": [ "Power Users", "Staff" ] }, { "Permission": "F5 SSL Profiles REST:CertificateStoreManagement_Schedule", "GrantedByRoles": [ "Power Users" ] }, ]` For more information about container permissions, see Container Permissions on page 629.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30.3 GET Security Identities Lookup

The GET /Security/Identities/Lookup method is used to confirm that the security identity specified is valid for the environment—the Active Directory forest in which Keyfactor Command is installed and any forests in a two-way trust (or one-way trust in a direction that allows the lookup to occur). It can be used to query an identity in the source identity store (Active Directory) to confirm its validity before using *POST /Security/Identities* (see POST Security Identities on page 2038) to create a new identity in Keyfactor Command with that user or group. This method returns HTTP 200 OK on a success with a response of true or false.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 566: GET Security Identities Lookup Input Parameters

Name	In	Description
Name	Query	**Required**. The identity name in the source identity store. For Active Directory users and groups, this can be given either as DOMAIN\name or name@-domain.com. For users in the local domain (the domain in which the Keyfactor Command server is installed), the lookup may be done without a domain name.

Table 567: GET Security Identities Lookup Response Data

Name	Description
Valid	A Boolean that indicates whether the provided name is valid (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30.4 GET Security Identities

The GET /Security/Identities method is used to return the list of security identities configured in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the security identities.

Note: This endpoint is for managing legacy formatted Active Directory identities only and is retained for backwards compatibility. New applications should use the *Security Claims* set of endpoints for both Active Directory and other identity providers (see Security on page 2027).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

Table 568: GET Security Identities Input Parameters

Name	In	Description
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*. *IdentityType* may be used as a sort order.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.
Validate	Query	A Boolean that specifies whether the optional parameter of *validate* is **false**, which allows the AuditXML validation to be skipped when loading records, or **true** (or not specified) in which case validation will occur. The default is **true**.

Table 569: GET Security Identities Response Data

Name	Description
Id	An integer containing the Keyfactor Command reference ID for the security identity.
AccountName	A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example: `KEYEXAMPLE\\PKI Administrators`
IdentityType	A string indicating the type of identity—User or Group.
Roles	An array of objects containing information about the security roles assigned to the security identity. Role information includes:

Name	In	Description
Id	Body	**Required**. An integer containing the Keyfactor Command identifier for the security role. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
Name	Body	**Required**. A string containing the short reference name for the security role.
Description	Body	**Required**. A string containing the description for the security role.
Enabled	Body	A Boolean that indicates whether the security role is enabled (true) or not (false). Security roles that have been disabled cannot be assigned to security identities. The default is *true*. This is considered deprecated and may be removed in a future release.
Immutable	Body	A Boolean that indicates whether the security role has been marked as editable (false) or not (true). Internal Keyfactor Command roles are not editable. This setting is reserved for Keyfactor Command internal use.
Valid	Body	A Boolean that indicates whether the security role's audit XML is valid (true) or not (false). A security role may become invalid if Keyfactor Command determines that it

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>appears to have been tampered with. This setting is not end-user configurable.</td></tr>
<tr><td>Private</td><td>Body</td><td>A Boolean that indicates whether the security role has been marked private (true) or not (false). The default is false.
This is considered deprecated and may be removed in a future release.</td></tr>
<tr><td>PermissionSetId</td><td>Body</td><td>A string indicating the Keyfactor Command reference GUID of the permission set associated with the role (see <a>Permission Sets on page 1959).</td></tr>
<tr><td>Identities</td><td>Body</td><td>An array of objects containing information about the security identities assigned to the security role. Identity details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer containing the Keyfactor Command identifier for the security identity.</td></tr>
<tr><td>AccountName</td><td>A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example:
KEYEXAMPLE\\PKI Administrators</td></tr>
<tr><td>IdentityType</td><td>A string indicating the type of identity—User or Group.</td></tr>
<tr><td>SID</td><td>A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.</td></tr>
</table>
</td></tr>
<tr><td>Permissions</td><td>Body</td><td>An array of strings containing the permissions assigned to</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>the role in a comma-separated list of Name:Value pairs. See Version One Permission Model on page 670 for an overview of the possible permissions.
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre></td></tr>
</table>
</td></tr>
<tr><td>Valid</td><td>A Boolean indicating whether the security identity's audit XML is valid (true) or not (false). A security identity may become invalid if Keyfactor Command determines that it appears to have been tampered with.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30.5 POST Security Identities

The POST /Security/Identities method is used to create a new security identity in Keyfactor Command. Use the *GET /Security/Identities/Lookup* method (see GET Security Identities Lookup on page 2033) before creating the new identity to confirm that the identity you plan to create is valid. This method returns HTTP 200 OK on a success with the details of the new security identity.

Note: This endpoint is for managing legacy formatted Active Directory identities only and is retained for backwards compatibility. New applications should use the *Security Claims* set of endpoints for both Active Directory and other identity providers (see Security on page 2027).

Tip: This method cannot be used to assign roles to an identity. Use the *PUT /Security/Roles* method (see PUT Security Roles on page 2105) to assign roles to an identity.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 570: POST Security Identities Input Parameters

Name	In	Description
AccountName	Body	**Required**. A string containing the account name for the security identity. For Active Directory user and groups, this will be in the form DOMAIN\\user or group name. For example: KEYEXAMPLE\\PKI Administrators

Table 571: POST Security Identities Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security identity.
AccountName	A string containing the account name for the security identity. For Active Directory user and groups, this will be in the form DOMAIN\\user or group name. For example: KEYEXAMPLE\\PKI Administrators
IdentityType	A string indicating the type of identity—User or Group.
Roles	An array of objects containing information about the security roles assigned to the security identity. For new security identities, this will be blank.
Valid	A Boolean that indicates whether the security identity's audit XML is valid (true) or not (false). A security identity may become invalid if Keyfactor Command determines that it appears to have been tampered with.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30.6 GET Security Containers ID Roles

The GET /Security/Containers/{id}/Roles method is used to return the list of security roles and permissions defined for the specified certificate store container. This method returns HTTP 200 OK on a success with details of the security roles and permissions for the container.

See also GET Security Roles ID Permissions Containers on page 2069 to list permissions on certificate store containers for a specified security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 572: GET Security Containers {id} Roles Input Parameters

Name	In	Description
id	Path	An integer indicating the Keyfactor Command reference ID of the certificate store container for which to retrieve permission information. Use the *GET /CertificateStoreContainers* method (see GET Certificate Store Containers on page 1477) to determine the ID of the certificate store container you wish to evaluate.

Table 573: GET Security Containers {id} Roles Response Data

Name	Description
SecurityRoleId	An integer indicating the Keyfactor Command reference ID of the security role granted permissions to the certificate store container.
Name	A string containing the short reference name for the security role granted permissions to the certificate store container.
Permissions	A comma-delimited array of strings indicating the permissions granted to the role for the certificate store container. See Certificate Store Management on page 673 for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30.7 POST Security Containers ID Roles

The POST /Security/Containers/{id}/Roles method is used to assign permissions for a security role to a certificate store container. This method returns HTTP 200 OK on a success with the details of the security role and permissions.

See also POST Security Roles ID Permissions Containers on page 2070 to assign permissions for one or more certificate store containers to a security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 574: POST Security Containers {id} Roles Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the certificate store container to update. Use the *GET /CertificateStoreContainers* method (see GET Certificate Store Containers on page 1477) to determine the ID of the certificate store container.
SecurityRoleId	Body	**Required**. An integer indicating the Keyfactor Command reference ID of the security role granted permissions to the certificate store container. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to determine the ID of the security role.
Permissions	Body	**Required**. A comma-delimited array of strings indicating the permissions granted to the role for the certificate store container. See Certificate Store Management on page 673 for an overview of the possible permissions. For example: `"Permissions": [ "Read", "Modify" ]`

Table 575: POST Security Containers {id} Roles Response Data

Name	Description
SecurityRoleId	An integer indicating the Keyfactor Command reference ID of the security role granted permissions to the certificate store container.
Name	A string containing the short reference name for the security role granted permissions to the certificate store container.
Permissions	A comma-delimited array of strings indicating the permissions granted to the role for the certificate store container. See Certificate Store Management on page 673 for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30.8 GET Security Audit Collections ID

The GET /Security/Audit/Collections/{id} method is used to return the list of security roles and permissions defined for the specified certificate collection. This method returns HTTP 200 OK on a success with details of the security roles and permissions for the collection.

See also GET Security Roles ID Permissions Collections on page 2074 to list permissions on certificate collections for a specified security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 576: GET Security Audit Collections {id} Input Parameters

Name	In	Description
id	Path	An integer indicating the Keyfactor Command reference ID of the certificate collection for which to retrieve permission information. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 1296) to determine the ID of the certificate collection you wish to evaluate.

Table 577: GET Security Audit Collections {id} Response Data

Name	Description
QueryId	An integer indicating the Keyfactor Command reference ID of the certificate collection.
AccessControlList	An array of objects containing the permissions granted to the user in a comma-separated list of arrays. See Version One Permission Model on page 670 for an overview of the possible permissions.

Name	Description
RoleId	An integer indicating the Keyfactor Command reference ID for the security role.
AreaPermissions	An array of comma-delimited integers indicating the collection permissions assigned to the role. System-Wide collection permissions set on a Role will not show as integers in the AreaPermission parameter. They will show as permissions in the Certificate > Collections section of the global permissions for that Role.

Integer	Area Permission
4	Collection Read
5	Collection Edit Metadata
7	Collection Download with Private Key
8	Collection Revoke
41	Collection Delete

For example:

```
"AccessControlList": [
   {
      "RoleId": "1",
      "AreaPermissions": [
         4,
         5,
         41
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

<pre>
      ]
   },
   {
      "RoleId": "3",
      "AreaPermissions": [
         4,
         8,
         41
      ]
   }
],
</pre>

</td></tr>
<tr><td>AssignableRoles</td><td>An array of objects containing the security roles defined in Keyfactor Command. Role information includes:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RoleId</td><td>An integer indicating the Keyfactor Command reference ID for the security role.</td></tr>
<tr><td>Name</td><td>A string containing the short reference name for the security role.</td></tr>
</table>

For example:

<pre>
"AssignableRoles": [
   {
      "RoleId": "1",
      "Name": "Administrator"
   },
   {
      "RoleId": "2",
      "Name": "Reporting API Access"
   }
],
</pre>

</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.30.9 GET Security My

The GET /Security/My method is used to return the list of security roles and permissions configured in Keyfactor Command for the current user. This method returns HTTP 200 OK on a success with the details of the security roles and permissions. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
None

Table 578: GET Security My Roles Response Data

Name	Description
Roles	An array of strings indicating the roles that the user holds.
GlobalPermissions	An array of objects containing the permissions granted to the user. See Version One Permission Model on page 670 for an overview of the possible permissions. For example: `"GlobalPermissions": [ { "Area": "AdminPortal", "Permission": "Read" }, { "Area": "Dashboard", "Permission": "Read" } ],`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.31 Security Claims

The Security Roles component of the Keyfactor API includes methods necessary to list, add, update, and delete security claims which are used to control user access to all aspects of Keyfactor Command.

Table 579: Security Claims Endpoints

Endpoint	Method	Description	Link
/{id}	GET	Returns details for the security claim with the specified ID, including permissions granted to the role and claims assigned the role.	GET Security Claims ID on page 2056
/	GET	Returns all security claims with filtering and output options.	GET Security Claims below
/	POST	Adds a new security claim.	POST Security Claims on page 2049
/	PUT	Updates the security claim with the specified ID.	PUT Security Claims on page 2053
/{id}	DELETE	Deletes the security claim with the specified ID.	DELETE Security Claims ID on page 2058
/Roles	GET	Returns the security claim assigned to the security claim identified by the selected parameters.	GET Security Claims Roles on page 2058

3.6.31.1 GET Security Claims

The GET /Security/Claims method is used to return the list of security claims configured in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the security claims.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 580: GET Security Claims Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Security Claim Search Feature on page 698. The query fields supported for this endpoint are: • ADClaimValue • ClaimType • ClaimValue • Description
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Provider*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 581: GET Security Claims Response Data

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td></td><td>An integer containing the Keyfactor Command reference ID for the security claim.</td></tr>
<tr><td>Description</td><td>Body</td><td>A string indicating a description for the security claim.</td></tr>
<tr><td>ClaimType</td><td>Body</td><td>A string indicating the type of claim. Supported values are:
<table>
<tr><th>Claim Type Integer</th><th>Claim Type String</th><th>Description</th></tr>
<tr><td>0</td><td>User</td><td>Active Directory user account</td></tr>
<tr><td>1</td><td>Group</td><td>Active Directory group.</td></tr>
<tr><td>2</td><td>Computer</td><td>Active Directory machine account</td></tr>
<tr><td>3</td><td>OAuthOid</td><td>An open authorization claim of a type not covered by client, role or subject</td></tr>
<tr><td>4</td><td>OAuthRole</td><td>An open authorization group claim</td></tr>
<tr><td>5</td><td>OAuthSubject</td><td>An open authorization user claim</td></tr>
<tr><td>6</td><td>OAuthClientId</td><td>An open authorization client application claim</td></tr>
</table></td></tr>
<tr><td>ClaimValue</td><td>Body</td><td>A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr>
<tr><td>Provider</td><td>Body</td><td>An object containing information about the provider assigned to the security claim.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor</td></tr>
</table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>Command reference GUID for the provider.</td></tr><tr><td>AuthenticationScheme</td><td>A string indicating the provider auth scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr><tr><td>DisplayName</td><td>A string containing the short reference name for the provider (e.g. Active Directory).</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.31.2 POST Security Claims

The POST /Security/Claims method is used to create a new security claim in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the new security claim.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 582: POST Security Claims Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Description</td><td>Body</td><td>Required. A string indicating a description for the security claim.</td></tr>
<tr><td>ClaimType</td><td>Body</td><td>Required. A string indicating the type of claim. Supported values are:
<table>
<tr><th>Claim Type Integer</th><th>Claim Type String</th><th>Description</th></tr>
<tr><td>0</td><td>User</td><td>Active Directory user account</td></tr>
<tr><td>1</td><td>Group</td><td>Active Directory group.</td></tr>
<tr><td>2</td><td>Computer</td><td>Active Directory machine account</td></tr>
<tr><td>3</td><td>OAuthOid</td><td>An open authorization claim of a type not covered by client, role or subject</td></tr>
<tr><td>4</td><td>OAuthRole</td><td>An open authorization group claim</td></tr>
<tr><td>5</td><td>OAuthSubject</td><td>An open authorization user claim</td></tr>
<tr><td>6</td><td>OAuthClientId</td><td>An open authorization client application claim</td></tr>
</table></td></tr>
<tr><td>ClaimValue</td><td>Body</td><td>Required. A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr>
</table>

Name	In	Description
ProviderAuthenticationScheme	Body	A string indicating the provider authentication scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).

Table 583: POST Security Claims Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer containing the Keyfactor Command reference ID for the security claim.</td></tr>
<tr><td>Description</td><td>A string indicating a description for the security claim.</td></tr>
<tr><td>ClaimType</td><td>A string indicating the type of claim. Supported values are:
<table>
<tr><th>Claim Type Integer</th><th>Claim Type String</th><th>Description</th></tr>
<tr><td>0</td><td>User</td><td>Active Directory user account</td></tr>
<tr><td>1</td><td>Group</td><td>Active Directory group.</td></tr>
<tr><td>2</td><td>Computer</td><td>Active Directory machine account</td></tr>
<tr><td>3</td><td>OAuthOid</td><td>An open authorization claim of a type not covered by client, role or subject</td></tr>
<tr><td>4</td><td>OAuthRole</td><td>An open authorization group claim</td></tr>
<tr><td>5</td><td>OAuthSubject</td><td>An open authorization user claim</td></tr>
<tr><td>6</td><td>OAuthClientId</td><td>An open authorization client application claim</td></tr>
</table>
</td></tr>
<tr><td>ClaimValue</td><td>A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr>
<tr><td>Provider</td><td>An object containing information about the provider assigned to the security claim.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID for the provider.</td></tr>
<tr><td>AuthenticationScheme</td><td>A string indicating the provider auth scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr>
<tr><td>DisplayName</td><td>A string containing the short reference name for the provider (e.g. Active Directory).</td></tr>
</table>
</td></tr>
<tr><td>ProviderAuthenticationScheme</td><td>A string indicating the provider authentication scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.31.3 PUT Security Claims

The PUT /Security/Claims method is used to update a security claim in Keyfactor Command. Only the claim description is editable for an existing claim. This method returns HTTP 200 OK on a success with the details of the security claim.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 584: PUT Security Claims Input Parameters

Name	Description
Id	An integer containing the Keyfactor Command reference ID for the security claim.
Description	A string indicating a description for the security claim.

Table 585: PUT Security Claims Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer containing the Keyfactor Command reference ID for the security claim.</td></tr>
<tr><td>Description</td><td>A string indicating a description for the security claim.</td></tr>
<tr><td>ClaimType</td><td>A string indicating the type of claim. Supported values are:
<table>
<tr><th>Claim Type Integer</th><th>Claim Type String</th><th>Description</th></tr>
<tr><td>0</td><td>User</td><td>Active Directory user account</td></tr>
<tr><td>1</td><td>Group</td><td>Active Directory group.</td></tr>
<tr><td>2</td><td>Computer</td><td>Active Directory machine account</td></tr>
<tr><td>3</td><td>OAuthOid</td><td>An open authorization claim of a type not covered by client, role or subject</td></tr>
<tr><td>4</td><td>OAuthRole</td><td>An open authorization group claim</td></tr>
<tr><td>5</td><td>OAuthSubject</td><td>An open authorization user claim</td></tr>
<tr><td>6</td><td>OAuthClientId</td><td>An open authorization client application claim</td></tr>
</table></td></tr>
<tr><td>ClaimValue</td><td>A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr>
<tr><td>Provider</td><td>An object containing information about the provider assigned to the security claim.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID for the provider.</td></tr>
<tr><td>AuthenticationScheme</td><td>A string indicating the provider auth scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr>
<tr><td>DisplayName</td><td>A string containing the short reference name for the provider (e.g. Active Directory).</td></tr>
</table></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.31.4 GET Security Claims ID

The GET /Security/Claims/{id} method is used to return a security claim by ID. This method returns HTTP 200 OK on a success with details for the specified security claim.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 586: GET Security Claims{id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the security claim to retrieve. Use the *GET /Security/Claims* method (see GET Security Claims on page 2046) to retrieve a list of all the security claims to determine the claim's ID.

Table 587: GET Security Claims{id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer containing the Keyfactor Command reference ID for the security claim.</td></tr>
<tr><td>Description</td><td>A string indicating a description for the security claim.</td></tr>
<tr><td>ClaimType</td><td>A string indicating the type of claim. Supported values are:
<table>
<tr><th>Claim Type Integer</th><th>Claim Type String</th><th>Description</th></tr>
<tr><td>0</td><td>User</td><td>Active Directory user account</td></tr>
<tr><td>1</td><td>Group</td><td>Active Directory group.</td></tr>
<tr><td>2</td><td>Computer</td><td>Active Directory machine account</td></tr>
<tr><td>3</td><td>OAuthOid</td><td>An open authorization claim of a type not covered by client, role or subject</td></tr>
<tr><td>4</td><td>OAuthRole</td><td>An open authorization group claim</td></tr>
<tr><td>5</td><td>OAuthSubject</td><td>An open authorization user claim</td></tr>
<tr><td>6</td><td>OAuthClientId</td><td>An open authorization client application claim</td></tr>
</table></td></tr>
<tr><td>ClaimValue</td><td>A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr>
<tr><td>Provider</td><td>An object containing information about the provider assigned to the security claim.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID for the provider.</td></tr>
<tr><td>AuthenticationScheme</td><td>A string indicating the provider auth scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr>
<tr><td>DisplayName</td><td>A string containing the short reference name for the provider (e.g. Active Directory).</td></tr>
</table></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.31.5 DELETE Security Claims ID

The DELETE /Security/Claims/{id} method is used to delete the security claim with the specified ID. This endpoint returns 204 with no content upon success.

Note: You cannot delete a claim that is used to grant permissions—via roles—to the user executing the delete command.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 588: DELETE Security Claims{id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the security claim that should be deleted from Keyfactor Command. Use the *GET /Security/Claims* method (see GET Security Claims on page 2046) to determine the ID of the security claim you wish to delete.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.31.6 GET Security Claims Roles

The GET /Security/Claims/Roles method is used to return the security roles assigned to the security claim identified by the selected parameters. Run GET Security Claims ID on page 2056 to determine the parameter values. This method returns HTTP 200 OK on a success with the details of the roles assigned to the claim.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 589: GET Security Claims Roles Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ClaimType</td><td>Body</td><td>Required. An integer indicating the type of claim. Supported values are:
<table>
<tr><th>Claim Type Integer</th><th>Claim Type String</th><th>Description</th></tr>
<tr><td>0</td><td>User</td><td>Active Directory user account</td></tr>
<tr><td>1</td><td>Group</td><td>Active Directory group.</td></tr>
<tr><td>2</td><td>Computer</td><td>Active Directory machine account</td></tr>
<tr><td>3</td><td>OAuthOid</td><td>An open authorization claim of a type not covered by client, role or subject</td></tr>
<tr><td>4</td><td>OAuthRole</td><td>An open authorization group claim</td></tr>
<tr><td>5</td><td>OAuthSubject</td><td>An open authorization user claim</td></tr>
<tr><td>6</td><td>OAuthClientId</td><td>An open authorization client application claim</td></tr>
</table></td></tr>
<tr><td>ClaimValue</td><td>Body</td><td>Required. A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr>
<tr><td>ProviderAuthenticationScheme</td><td>Body</td><td>Required. A string indicating the provider authentication scheme (e.g. Active Directory, or Client Certificate</td></tr>
</table>

Name	In	Description
		Authentication CA, or unknown). The value in the *Provider: AuthenticationScheme:* parameter from GET Security Claims on page 2046 or GET Security Claims ID on page 2056.

Table 590: GET Security Claims Roles Response Data

Name	Description
Id	An integer containing the Keyfactor Command reference ID for the security role.
Name	A string containing the short reference name of the security role.
Description	A string containing the description of the security role.
PermissionSetId	A string containing the Keyfactor Command reference GUID for the permission set assigned to the security role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32 Security Roles Permissions

The Security Roles Permissions component of the Keyfactor API includes methods necessary to list, add, and update security roles permissions at the role, global, provider, container and collection-level.

Table 591: Security Roles Permissions Endpoints

Endpoint	Method	Description	Link
/{id}/Permisssions	GET	Returns all permissions associated with the security role that matches the id	GET Security Roles ID Permissions on the next page
/{id}/Permisssions/Global	GET	Returns all global permissions associated with the security role that matches the ID.	GET Security Roles ID Permissions Global on page 2064
/{id}/Permisssions/Global	POST	Adds global permissions to the security role that matches the id. Note that the Areas *Certificates* and *CertificateStoreManagement* are reserved for collection and container permissions, respectively.	POST Security Roles ID Permissions Global on page 2065
/{id}/Permisssions/Global	PUT	Sets global permissions of the security role that matches the ID. Note that the Areas *Certificates* and *CertificateStoreManagement* are reserved for collection and container permissions, respectively.	PUT Security Roles ID Permissions Global on page 2067
/{id}/Permisssions/Containers	GET	Returns all container permissions associated with the security role that matches the ID.	GET Security Roles ID Permissions Containers on page 2069
/{id}/Permisssions/Containers	POST	Adds container permissions to the security role that matches the ID.	POST Security Roles ID Permissions Containers on page 2070
/{id}/Permisssions/Containers	PUT	Sets container permissions to the security role that matches the ID.	PUT Security Roles ID Permissions

Endpoint	Method	Description	Link
			Containers on page 2072
/{id}/Permisssions/Collections	GET	Returns all collection permissions associated with the security role that matches the ID.	GET Security Roles ID Permissions Collections on page 2074
/{id}/Permisssions/Collections	POST	Adds collection permissions to the security role that matches the ID.	POST Security Roles ID Permissions Collections on page 2075
/{id}/Permisssions/Collections	PUT	Sets collection permissions to the security role that matches the ID.	PUT Security Roles ID Permissions Collections on page 2077
/{id}/Permissions/PamProviders	GET	Returns all PAM provider permissions associated with the security role that matches the ID.	GET Security Roles ID Permissions PAM Providers on page 2079
/{id}/Permissions/PamProviders	PUT	Sets PAM provider permissions to the security role that matches the ID.	PUT Security Roles ID Permissions PAM Providers on page 2080

3.6.32.1 GET Security Roles ID Permissions

The GET /Security/Roles/{id}/Permissions method is used to return all permissions associated with the security role that matches the ID. This method returns HTTP 200 OK on a success with certificate store container permission details for the specified security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 592: GET Security Roles {id} Permissions Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to retrieve permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 593: GET Security Roles {id} Permissions Response Data

Name	Description
	An object containing information about the permissions granted to the security role. Details include:

Name	Description
Type	A string containing the area at which the permission is applied to (global, container, or collection).
Area	A string containing the name of the permission (e.g. Certificates).
Permission	A string indicating the permission level granted in the area for this role (e.g. Read).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.2 GET Security Roles ID Permissions Global

The GET /Security/Roles/{id}/Permissions/Global method is used to return all global permissions associated with the security role that matches the ID. This method returns HTTP 200 OK on a success with certificate store container permission details for the specified security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 594: GET Security Roles {id} Global Permissions Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to retrieve global permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 595: GET Security Roles {id} Global Permissions Response Data

Name	Description
	An object containing information about the global permissions granted to the security role. Details include:

Name	Description
Area	A string containing the name of the permission (e.g. Certificates).
Permission	A string indicating the permission level granted in the area for this role (e.g. Read).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.3 POST Security Roles ID Permissions Global

The POST /Security/Roles/{id}/Permissions/Global method is used to add global permissions to the security role that matches the ID. This method returns HTTP 200 OK on a success with global permission details for the specified security role.

Important: Only the permission settings included in the command will be affected. Any other permissions settings will not be affected and remain as is.

Note: The API Endpoint utility header displays a list of valid global Area and Permission combinations.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 596: POST Security Roles {id}Global Permissions Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to set global permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
globalPermissions	Body	An object containing information about the global permissions granted for this security role. **Note:** See the API endpoint header for a list of all the valid Area and Permission combinations. Details include: Name: Area — **Required**. A string indicating the name of the permissions to grant (e.g. AdminPortal). Name: Permission — **Required**. A string indicating the permission level to grant (e.g. Read)

Table 597: POST Security Roles {id} Global Permissions Response Data

Name	Description
	An object containing information about the global permissions granted to the security role. Details include: Name: Area — A string containing the name of the permission (e.g. Certificates). Name: Permission — A string indicating the permission level granted in the area for this role (e.g. Read).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.4 PUT Security Roles ID Permissions Global

The PUT /Security/Roles/{id}/Permissions/Global method is used to update the global permissions granted to the specified security role by ID. Note that the areas *Certificates* and *CertificateStoreManagement* are reserved for collection and container permissions. This method returns HTTP 200 OK on a success with global permission details for the specified security role.

Important: Any previously defined permissions of the given type (e.g. global) that are not submitted with their full existing data using this method will be cleared of their existing data. Existing data for other types will be retained. When using this method, you should first do a GET to retrieve all the permissions of the given type for the role you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing. If you just wish to add permissions without modifying existing permissions, use the POST method.

Note: The API Endpoint utility displays a list of valid global permissions on the endpoint.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 598: PUT Security Roles {id}Global Permissions Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to set global permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
globalPermissions	Body	An object containing information about the global permissions granted for this security role. **Note:** See the API endpoint header for a list of all the valid Area and Permission combinations. Details include: Name: Area — **Required**. A string indicating the name of the permissions to grant (e.g. AdminPortal). Name: Permission — **Required**. A string indicating the permission level to grant (e.g. Read)

Table 599: PUT Security Roles {id} Global Permissions Response Data

Name	Description
	An object containing information about the global permissions granted to the security role. Details include: Name: Area — A string containing the name of the permission (e.g. Certificates). Name: Permission — A string indicating the permission level granted in the area for this role (e.g. Read).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.5 GET Security Roles ID Permissions Containers

The GET /Security/Roles/{id}/Permissions/Containers method is used to return all certificate store container permissions associated with the security role that matches the ID. This method returns HTTP 200 OK on a success with certificate store container permission details for the specified security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 600: GET Security Roles {id} Permissions Containers Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to retrieve certificate store container permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 601: GET Security Roles {id} Permissions Containers Response Data

Name	Description
	An object containing information about the certificate store container permissions granted to the security role. Details include:

Name	Description
ContainerId	An integer containing the container ID.
Name	A string containing the name of the certificate store container.
Permission	A string indicating the permission granted on the entity for this role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.6 POST Security Roles ID Permissions Containers

The POST /Security/Roles/{id}/Permissions/Containers method is used to add new container permissions to the security role that matches the ID. This method returns HTTP 200 OK on a success with certificate store container permission details for the specified security role.

Important: Only the permission settings included in the command will be affected. Any other permissions settings will not be affected and remain as is.

Note: The API Endpoint utility displays a list of valid global permissions on the endpoint.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 602: POST Security Roles {id} Permissions Containers Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to set certificate store container permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
containerPermissions	Body	An object containing information about the permissions granted to certificate store containers for this security role. Container details include: **Name** / **Description** **ContainerId** — **Required**. An integer containing the Keyfactor Command identifier for the certificate store container. **Permission** — **Required**. A string indicating the permission granted on the container for this role—*Read*, *Schedule*, or *Modify*. **Tip:** Users with *Modify* permissions on a container inherit *Read* and *Schedule*; users with *Schedule* permissions on a container inherit *Read*.

Table 603: POST Security Roles {id} Permissions Containers Response Data

Name	Description
	An object containing information about the certificate store container permissions granted to the security role. Details include: **Name** / **Description** **ContainerId** — An integer containing the container ID. **Name** — A string containing the name of the certificate store container. **Permission** — A string indicating the permission granted on the entity for this role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.7 PUT Security Roles ID Permissions Containers

The PUT /Security/Roles/{id}/Permissions/Containers method is used to update container permissions to the security role that matches the ID. This method returns HTTP 200 OK on a success with certificate store container permission details for the specified security role.

Important: Any previously defined permissions of the given type (e.g. container) that are not submitted with their full existing data using this method will be cleared of their existing data. Existing data for other types will be retained. When using this method, you should first do a GET to retrieve all the permissions of the given type for the role you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing. If you just wish to add permissions without modifying existing permissions, use the POST method.

Note: The API Endpoint utility displays a list of valid global permissions on the endpoint.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 604: PUT Security Roles {id} Permissions Containers Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to set certificate store container permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
containerPermissions	Body	An object containing information about the permissions granted to certificate store containers for this security role. Container details include: **Name** / **Description** **ContainerId** — **Required**. An integer containing the Keyfactor Command identifier for the certificate store container. **Permission** — **Required**. A string indicating the permission granted on the container for this role—*Read*, *Schedule*, or *Modify*. **Tip:** Users with *Modify* permissions on a container inherit *Read* and *Schedule*; users with *Schedule* permissions on a container inherit *Read*.

Table 605: PUT Security Roles {id} Permissions Containers Response Data

Name	Description
	An object containing information about the certificate store container permissions granted to the security role. Details include: **Name** / **Description** **ContainerId** — An integer containing the container ID. **Name** — A string containing the name of the certificate store container. **Permission** — A string indicating the permission granted on the entity for this role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.8 GET Security Roles ID Permissions Collections

The GET /Security/Roles/{id}/Permissions/Collections method is used to return all certificate collection permissions associated with the security role that matches the ID. This method returns HTTP 200 OK on a success with certificate collection permission details for the specified security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 606: GET Security Roles {id} Permissions Collections Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to retrieve certificate collection permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 607: GET Security Roles {id} Permissions Collections Response Data

Name	Description
	An object containing information about the certificate collection permissions granted to the security role. Details include:

Name	Description
CollectionId	An integer containing the collection ID.
Name	A string containing the name of the certificate collection .
Permission	A string indicating the permission granted on the entity for this role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.9 POST Security Roles ID Permissions Collections

The POST/Security/Roles/{id}/Permissions/Collections method is used to add new collection permissions to the security role that matches the ID. This method returns HTTP 200 OK on a success with certificate collection permission details for the specified security role.

Important: Only the permission settings included in the command will be affected. Any other permissions settings will not be affected and remain as is.

Note: The API Endpoint utility displays a list of valid global permissions on the endpoint.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 608: POST Security Roles {id} Permissions Collections Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to set certificate collection permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
collectionPermissions	Body	An object containing information about the permissions granted to certificate collection for this security role. Collection details include: **Name** / **Description** CollectionId — **Required**. An integer containing the Keyfactor Command identifier for the certificate collection. Permission — **Required**. A string indicating the permission granted on the collection for this role—*Read*, *EditMetadata*, *Recover*, *Revoke*, or *Delete*.

Table 609: POST Security Roles {id} Permissions Collections Response Data

Name	Description
	An object containing information about the certificate collection permissions granted to the security role. Details include: **Name** / **Description** CollectionId — An integer containing the collection ID. Name — A string containing the name of the certificate collection . Permission — A string indicating the permission granted on the entity for this role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.10 PUT Security Roles ID Permissions Collections

The PUT /Security/Roles/{id}/Permissions/Collections method is used to update collection permissions to the security role that matches the ID. It replaces the deprecated endpoint: POST /CertificateCollections/{id}/Permissions. This method returns HTTP 200 OK on a success with certificate collection permission details for the specified security role.

Important: Any previously defined permissions of the given type (e.g. collection) that are not submitted with their full existing data using this method will be cleared of their existing data. Existing data for other types will be retained. When using this method, you should first do a GET to retrieve all the permissions of the given type for the role you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing. If you just wish to add permissions without modifying existing permissions, use the POST method.

Note: The API Endpoint utility displays a list of valid global permissions on the endpoint.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 610: PUT Security Roles {id} Permissions Collections Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to set certificate collection permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
collectionPermissions	Body	An object containing information about the permissions granted to certificate collection for this security role. Collection details include: **Name** / **Description** CollectionId — **Required**. An integer containing the Keyfactor Command identifier for the certificate collection. Permission — **Required**. A string indicating the permission granted on the collection for this role—*Read*, *EditMetadata*, *Recover*, *Revoke*, or *Delete*.

Table 611: PUT Security Roles {id} Permissions Collections Response Data

Name	Description
	An object containing information about the certificate collection permissions granted to the security role. Details include: **Name** / **Description** CollectionId — An integer containing the collection ID. Name — A string containing the name of the certificate collection . Permission — A string indicating the permission granted on the entity for this role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.11 GET Security Roles ID Permissions PAM Providers

The GET /Security/Roles/{id}/Permissions/PamProviders method is used to return all PAM provider permissions associated with the security role with the specified ID. This method returns HTTP 200 OK on a success with PAM provider permission details for the specified security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Table 612: GET Security Roles {id} Permissions PAM Providers Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the security role for which to retrieve PAM provider permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 613: GET Security Roles {id} Permissions PAM Providers Response Data

Name	Description
	An object containing information about the certificate permissions granted to the security role. Details include: Name / Description Id — An integer containing the ID. Name — A string containing the name of the certificate . Permission — A string indicating the permission granted on the entity for this role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.32.12 PUT Security Roles ID Permissions PAM Providers

The PUT /Security/Roles/{id}/Permissions/PamProviders method is used to update PAM provider permissions on the security role that matches the specified ID. This method returns HTTP 200 OK on a success with PAM provider permission details for the specified security role.

Important: Any previously defined permissions of the given type (e.g.) that are not submitted with their full existing data using this method will be cleared of their existing data. Existing data for other types will be retained. When using this method, you should first do a GET to retrieve all the permissions of the given type for the role you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing. If you just wish to add permissions without modifying existing permissions, use the POST method.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

Table 614: PUT Security Roles {id} Permissions PAM Providers Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID of the security role for which to set permissions. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 615: PUT Security Roles {id} Permissions PAM Providers Response Data

Name	Description
	An object containing information about the certificate permissions granted to the security role. Details include: Name / Description Id — An integer containing the ID. Name — A string containing the name of the certificate . Permission — A string indicating the permission granted on the entity for this role.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.33 Security Roles

The Security Roles component of the Keyfactor API includes methods necessary to list, add, update, and delete security roles which are used to control access to all aspects of Keyfactor Command.

Table 616: Security Roles Endpoints

Endpoint	Method	Description	Link
/{id}	GET	Returns details for the security role with the specified ID, including permissions granted to the role and claims assigned the role.	GET Security Roles ID on the next page
/{id}	DELETE	Deletes the security role with the specified ID.	DELETE Security Roles ID on the next page
/	GET	Returns all security roles with filtering and output options.	GET Security Roles on page 2088
/	POST	Adds a new security role.	POST Security Roles on page 2093
/	PUT	Updates the security role with the specified ID.	PUT Security Roles on page 2105
/{id}/Identities	GET	Returns the security identities assigned to the security role with the specified ID.	GET Security Roles ID Identities on page 2121
/{id}/Identities	PUT	Updates the security identities assigned to the security role with the specified ID.	PUT Security Roles ID Identities on page 2120
/{id}/Copy	POST	Adds a new security role by copying the existing security role with the specified ID.	POST Security Roles ID Copy on page 2117

3.6.33.1 DELETE Security Roles ID

The DELETE /Security/Roles/{id} method is used to delete the security role with the specified ID. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to determine the ID of the security role you wish to delete. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Table 617: DELETE Security Roles {id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the security role that should be deleted from Keyfactor Command.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.33.2 GET Security Roles ID

The GET /Security/Roles/{id} method is used to return a security role by ID. This method returns HTTP 200 OK on a success with details for the specified security roles.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

This method has two available versions. Keyfactor strongly recommends using the newer method when possible; the v1 method has been deprecated since it supports Active Directory identities only. The v2 method has been redesigned to provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. This version of the method supports both Active Directory and other identity providers. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the GET /Security/Roles/{id} method has been redesigned to provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. All new development should use this version.

Table 618: GET Security Roles {id} v2 Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the security role to retrieve. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 619: GET Security Roles {id} v2 Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security role.
Name	A string containing the short reference name for the security role.
Description	A string containing the description for the security role.
Immutable	A Boolean indicating if the role is immutable or not. Only the built-in *Administrators* role is considered immutable. The value of this parameter cannot be changed.
PermissionSetId	A string containing the Keyfactor Command reference GUID of the permission set to which the role is assigned (see Permission Sets on page 1959).
Permissions	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Version Two Permission Model on page 632 for an overview of the possible permissions. For example: `"Permissions": [ "/portal/read/", "/dashboard/read/", "/certificates/collections/metadata/modify/6/", "/certificates/collections/private_key/read/6/" ],`
Claims	An array of objects containing the claims associated with the role.

Name	In	Description
Description	Body	A string indicating a description for the security claim.
ClaimType	Body	A string indicating the type of claim. Supported values are:

Claim Type Integer	Claim Type String	Description
0	User	Active

Name	Description

Name	In	Description

Claim Type Integer	Claim Type String	Description
		Directory user account
1	Group	Active Directory group.
2	Computer	Active Directory machine account
3	OAuthOid	An open authorization claim of a type not covered by client, role or subject
4	OAuthRole	An open authorization group claim
5	OAuthSubject	An open authorization user claim
6	OAuthClientId	An open authorization client application claim

Name	Description
	<table><tr><th>Name</th><th>In</th><th>Description</th></tr><tr><td>ClaimValue</td><td>Body</td><td>A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr><tr><td>Provider-AuthenticationScheme</td><td>Body</td><td>A string indicating the provider authentication scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr></table>

Version 1

Version 1 of the GET /Security/Roles/{id} method includes the same capabilities as version 2, but offers support for managing legacy formatted Active Directory identities.

Important: This has been deprecated since it supports Active Directory identities only. It is retained for backwards compatibility, but all new development should use methods that provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. These newer methods support both Active Directory and other identity providers. See version 2 of this method.

Table 620: GET Security Roles {id} v1 Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the security role to retrieve. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 621: GET Security Roles {id} v1 Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security role.
Name	A string containing the short reference name for the security role.
Description	A string containing the description for the security role.
Enabled	A Boolean that indicates whether the security role is enabled (true) or not (false). Security roles that have been disabled cannot be assigned to security identities. The default is *true*. This is considered deprecated and may be removed in a future release.
Immutable	A Boolean that indicates whether the security role has been marked as editable (false) or not (true). Internal Keyfactor Command roles are not editable. This setting is reserved for Keyfactor Command internal use.
Valid	A Boolean that indicates whether the security role's audit XML is valid (true) or not (false). A security role may become invalid if Keyfactor Command determines that it appears to have been tampered with. This setting is not end-user configurable.
Private	A Boolean that indicates whether the security role has been marked private (true) or not (false). The default is *false*. This is considered deprecated and may be removed in a future release.
PermissionSetId	A string indicating the Keyfactor Command reference GUID of the permission set associated with the role (see Permission Sets on page 1959).
Identities	An array of objects containing information about the security identities assigned to the security role. Identity details include:

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security identity.
AccountName	A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example: `KEYEXAMPLE\\PKI Administrators`
IdentityType	A string indicating the type of identity—User or Group.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SID</td><td>A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.</td></tr>
</table>
</td></tr>
<tr><td>Permissions</td><td>An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See <a>Version One Permission Model on page 670 for an overview of the possible permissions.
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.33.3 GET Security Roles

The GET /Security/Roles method is used to return the list of security roles configured in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the security roles.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

This method has two available versions. Keyfactor strongly recommends using the newer method when possible; the v1 method has been deprecated since it supports Active Directory identities only. The v2 method has been redesigned to provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. This version of the method supports both Active Directory and other identity providers. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the GET /Security/Roles method has been redesigned to provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. All new development should use this version.

Table 622: GET Security Roles v2 Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Security Role Search Feature on page 690. The query fields supported for this endpoint are: • Name
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 623: GET Security Roles v2 Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security role.
Name	A string containing the short reference name for the security role.
Immutable	A Boolean indicating if the role is immutable or not. Only the built-in *Administrators* role is considered immutable. The value of this parameter cannot be changed.
PermissionSetId	A string containing the Keyfactor Command reference GUID of the permission set to which the role is assigned (see Permission Sets on page 1959). **Tip:** For details of the permissions associated with the role, use the *GET /SecurityRoles/{id}* method (see GET Security Roles ID on page 2082) for the desired role. For details of the permissions associated withe the permission set to which the role belongs, use *GET /PermissionSets/{id}* method (see GET Permission Sets ID on page 1961) using this permission set GUID. A role may be only granted a subset of the permissions available in a permission set.
ClaimsCount	An integer indicating the number of claims mapped to the role. **Tip:** For details of the claims associated with the role, use the *GET /SecurityRoles/{id}* method (see GET Security Roles ID on page 2082) for the desired role.

Version 1

Version 1 of the GET /Security/Roles method includes the same capabilities as version 2, but offers support for managing legacy formatted Active Directory identities only.

Important: This has been deprecated since it supports Active Directory identities only. It is retained for backwards compatibility, but all new development should use methods that provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. These newer methods support both Active Directory and other identity providers. See version 2 of this method.

Table 624: GET Security Roles v1 Input Parameters

Name	In	Description
Validate	Query	A Boolean that specifies whether the optional parameter of *validate* is **false**, which allows the AuditXML validation to be skipped when loading records, or **true** (or not specified) in which case validation will occur. The default is **true**.
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Security Role Search Feature on page 690. The query fields supported for this endpoint are: • Name
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
SkipCount	Query	An integer indicating the number of records that should be skipped in providing results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 625: GET Security Roles v1 Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security role.
Name	A string containing the short reference name for the security role.
Description	A string containing the description for the security role.
Enabled	A Boolean that indicates whether the security role is enabled (true) or not (false). Security roles that have been disabled cannot be assigned to security identities. The default is *true*. This is considered deprecated and may be removed in a future release.
Immutable	A Boolean that indicates whether the security role has been marked as editable (false) or not (true). Internal Keyfactor Command roles are not editable. This setting is reserved for Keyfactor Command internal use.
Valid	A Boolean that indicates whether the security role's audit XML is valid (true) or not (false). A security role may become invalid if Keyfactor Command determines that it appears to have been tampered with. This setting is not end-user configurable.
Private	A Boolean that indicates whether the security role has been marked private (true) or not (false). The default is *false*. This is considered deprecated and may be removed in a future release.
PermissionSetId	A string indicating the Keyfactor Command reference GUID of the permission set associated with the role (see Permission Sets on page 1959).
Identities	An array of objects containing information about the security identities assigned to the security role. Identity details include:

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security identity.
AccountName	A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example: `KEYEXAMPLE\\PKI Administrators`
IdentityType	A string indicating the type of identity—User or Group.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SID</td><td>A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.</td></tr>
</table>
</td></tr>
<tr><td>Permissions</td><td>An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See <a>Version One Permission Model on page 670 for an overview of the possible permissions.
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.33.4 POST Security Roles

The POST /Security/Roles method is used to create a new security role in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

This method has two available versions. Keyfactor strongly recommends using the newer method when possible; the v1 method has been deprecated since it supports Active Directory identities only. The v2 method has been redesigned to provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. This version of the method supports both Active Directory and other identity providers. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the POST /Security/Roles method has been redesigned to provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. All new development should use this version.

Table 626: POST Security Roles v2 Input Parameters

Name	Description
Name	**Required**. A string containing the short reference name for the security role.
Description	**Required**. A string containing the description for the security role.
PermissionSetId	A string containing the Keyfactor Command reference GUID of the permission set to which the role is assigned (see Permission Sets on page 1959).
Permissions	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Version Two Permission Model on page 632 for an overview of the possible permissions. For example: `"Permissions": [ "/portal/read/", "/dashboard/read/", "/certificates/collections/metadata/modify/6/", "/certificates/collections/private_key/read/6/" ],`
Claims	An array of objects containing the claims associated with the role.

Name	In	Description
Description	Body	A string indicating a description for the security claim.
ClaimType	Body	A string indicating the type of claim. Supported values are:

Claim Type Integer	Claim Type String	Description
0	User	Active Directory user account
1	Group	Active

Name	Description

Name	In	Description

Claim Type Integer	Claim Type String	Description
		Directory group.
2	Computer	Active Directory machine account
3	OAuthOid	An open authorization claim of a type not covered by client, role or subject
4	OAuthRole	An open authorization group claim
5	OAuthSubject	An open authorization user claim
6	OAuthClientId	An open authorization client application claim

ClaimValue	Body	A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g.

Name	Description
	<table><tr><th>Name</th><th>In</th><th>Description</th></tr><tr><td></td><td></td><td>KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr><tr><td>Provider-AuthenticationScheme</td><td>Body</td><td>A string indicating the provider authentication scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr></table>

Table 627: POST Security Roles v2 Response Data

Name	In	Description
Id	Body	An integer containing the Keyfactor Command identifier for the security role.
Name	Body	A string containing the short reference name for the security role.
Description	Body	A string containing the description for the security role.
Immutable	Body	A Boolean indicating if the role is immutable (true) or not (false). Only the built-in *Administrators* role is considered immutable. The value of this parameter cannot be changed.
PermissionSetId	Body	A string containing the Keyfactor Command reference GUID of the permission set to which the role is assigned (see Permission Sets on page 1959).
Permissions	Body	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Version Two Permission Model on page 632 for an overview of the possible permissions. For example: `"Permissions": [ "/portal/read/", "/dashboard/read/", "/certificates/collections/metadata/modify/6/", "/certificates/collections/private_key/read/6/" ],`
Claims	Body	An array of objects containing the claims associated with the role.

Name	In	Description
Id		An integer containing the Keyfactor Command reference ID for the security claim.
Description	Body	A string indicating a description for the security claim.
ClaimType	Body	A string indicating the type of claim. Supported values are:

Name	In	Description
		Name / **In** / **Description** ClaimValue / Body / A string containing the identifying information for the entity specified in the claim. For Active

Claim Type Integer	Claim Type String	Description
0	User	Active Directory user account
1	Group	Active Directory group.
2	Computer	Active Directory machine account
3	OAuthOid	An open authorization claim of a type not covered by client, role or subject
4	OAuthRole	An open authorization group claim
5	OAuthSubject	An open authorization user claim
6	OAuthClientId	An open authorization client application claim

Name	In	Description
		<table><tr><th>Name</th><th>In</th><th>Description</th></tr><tr><td></td><td></td><td>Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr><tr><td>Provider</td><td>Body</td><td>An object containing information about the provider assigned to the security claim.<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID for the provider.</td></tr><tr><td>AuthenticationScheme</td><td>A string indicating the provider auth scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr><tr><td>DisplayName</td><td>A string containing the short reference name for the provider (e.g. Active Directory).</td></tr></table></td></tr></table>

Version 1

Version 1 of the POST /Security/Roles method includes the same capabilities as version 2, but offers support for managing legacy formatted Active Directory identities only.

Important: This has been deprecated since it supports Active Directory identities only. It is retained for backwards compatibility, but all new development should use methods that provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. These newer methods support both Active Directory and other identity providers. See version 2 of this method.

Table 628: POST Security Roles v1 Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string containing the short reference name for the security role.</td></tr>
<tr><td>Description</td><td>Body</td><td>Required. A string containing the description for the security role.</td></tr>
<tr><td>Enabled</td><td>Body</td><td>A Boolean that indicates whether the security role is enabled (true) or not (false). Security roles that have been disabled cannot be assigned to security identities. The default is true.
This is considered deprecated and may be removed in a future release.</td></tr>
<tr><td>Private</td><td>Body</td><td>A Boolean that indicates whether the security role has been marked private (true) or not (false). The default is false.
This is considered deprecated and may be removed in a future release.</td></tr>
<tr><td>PermissionSetId</td><td>Body</td><td>A string indicating the Keyfactor Command reference GUID of the permission set associated with the role (see <a>Permission Sets on page 1959).</td></tr>
<tr><td>Permissions</td><td>Body</td><td>An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See <a>Version One Permission Model on page 670 for an overview of the possible permissions.
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre></td></tr>
<tr><td>Identities</td><td>Body</td><td>An array of objects containing one or more identifiers for each security identity to associate with the role. Supported identifiers include:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>AccountName</td><td>Required*. A string containing the account name for the security identity. For Active Directory user and groups, this will be in the form DOMAIN\\user or group name. For example:
KEYEXAMPLE\\PKI Administrators
* One of AccountName or SID is required in</td></tr></table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>order to specify an identity, but not both.</td></tr><tr><td>SID</td><td>**Required**[*]. A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.
		[*] One of *AccountName* or *SID* is required in order to specify an identity, but not both.</td></tr></table>
		For example:
		<pre>"Identities": [
		  {
		  "AccountName": "KEYEXAMPLE\\jsmith"
		  },
		  {
		  "AccountName": "KEYEXAMPLE\\mjones"
		  }
		]</pre>

Table 629: POST Security Roles v1 Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security role.
Name	A string containing the short reference name for the security role.
Description	A string containing the description for the security role.
Enabled	A Boolean that indicates whether the security role is enabled (true) or not (false). Security roles that have been disabled cannot be assigned to security identities. The default is *true*. This is considered deprecated and may be removed in a future release.
Immutable	A Boolean that indicates whether the security role has been marked as editable (false) or not (true). Internal Keyfactor Command roles are not editable. This setting is reserved for Keyfactor Command internal use.
Valid	A Boolean that indicates whether the security role's audit XML is valid (true) or not (false). A security role may become invalid if Keyfactor Command determines that it appears to have been tampered with. This setting is not end-user configurable.
Private	A Boolean that indicates whether the security role has been marked private (true) or not (false). The default is *false*. This is considered deprecated and may be removed in a future release.
PermissionSetId	A string indicating the Keyfactor Command reference GUID of the permission set associated with the role (see Permission Sets on page 1959).
Identities	An array of objects containing information about the security identities assigned to the security role. Identity details include:

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security identity.
AccountName	A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example: `KEYEXAMPLE\\PKI Administrators`
IdentityType	A string indicating the type of identity—User or Group.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SID</td><td>A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.</td></tr>
</table>
</td></tr>
<tr><td>Permissions</td><td>An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See <a>Version One Permission Model on page 670 for an overview of the possible permissions.
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.33.5 PUT Security Roles

The PUT /Security/Roles method is used to update a security role in Keyfactor Command including the permissions set for the role and the security identities mapped to the role. This method returns HTTP 200 OK on a success with the details of the security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

This method has two available versions. Keyfactor strongly recommends using the newer method when possible; the v1 method has been deprecated since it supports Active Directory identities only. The v2 method has been redesigned to provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. This version of the method supports both Active Directory and other identity providers. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the PUT /Security/Roles method has been redesigned to provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. All new development should use this version.

Table 630: PUT Security Roles v2 Input Parameters

Name	In	Description
Id	Body	**Required**. An integer containing the Keyfactor Command identifier for the security role. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
Name	Body	**Required**. A string containing the short reference name for the security role.
Description	Body	**Required**. A string containing the description for the security role.
PermissionSetId	Body	A string containing the Keyfactor Command reference GUID of the permission set to which the role is assigned (see Permission Sets on page 1959).

Permissions	Body	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Version Two Permission Model on page 632 for an overview of the possible permissions.
		For example:
		`"Permissions": [`
		`"/portal/read/",`
		`"/dashboard/read/",`
		`"/certificates/collections/metadata/modify/6/",`
		`"/certificates/collections/private_key/read/6/"`
		`],`

Claims	Body	An array of objects containing the claims associated with the role.

Name	In	Description
Description	Body	A string indicating a description for the security claim.
ClaimType	Body	A string indicating the type of claim. Supported values are:

Name	In	Description
		Name / **In** / **Description** **Claim Type Integer** / **Claim Type String** / **Description** 0 / User / Active Directory user account 1 / Group / Active Directory group. 2 / Computer / Active Directory machine account 3 / OAuthOid / An open authorization claim of a type not covered by client, role or subject 4 / OAuthRole / An open authorization group claim 5 / OAuthSubject / An open authorization user claim

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Claim Type Integer</th><th>Claim Type String</th><th>Description</th></tr>
<tr><td>6</td><td>OAuthClientId</td><td>An open authorization client application claim</td></tr>
</table>
</td></tr>
<tr><td>ClaimValue</td><td>Body</td><td>A string containing the identifying information for the entity specified in the claim. For Active Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr>
<tr><td>ProviderAuthenticationScheme</td><td>Body</td><td>A string indicating the provider authentication scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr>
</table>
</td></tr>
</table>

Table 631: PUT Security Roles v2 Response Data

Name	In	Description
Id	Body	An integer containing the Keyfactor Command identifier for the security role.
Name	Body	A string containing the short reference name for the security role.
Description	Body	A string containing the description for the security role.
Immutable	Body	A Boolean indicating if the role is immutable (true) or not (false). Only the built-in *Administrators* role is considered immutable. The value of this parameter cannot be changed.
PermissionSetId	Body	A string containing the Keyfactor Command reference GUID of the permission set to which the role is assigned (see Permission Sets on page 1959).
Permissions	Body	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Version Two Permission Model on page 632 for an overview of the possible permissions. For example: `"Permissions": [ "/portal/read/", "/dashboard/read/", "/certificates/collections/metadata/modify/6/", "/certificates/collections/private_key/read/6/" ],`
Claims	Body	An array of objects containing the claims associated with the role.

Name	In	Description
Id		An integer containing the Keyfactor Command reference ID for the security claim.
Description	Body	A string indicating a description for the security claim.
ClaimType	Body	A string indicating the type of claim. Supported values are:

Name	In	Description
		<table><tr><th>Name</th><th>In</th><th>Description</th></tr><tr><td></td><td></td><td><table><tr><th>Claim Type Integer</th><th>Claim Type String</th><th>Description</th></tr><tr><td>0</td><td>User</td><td>Active Directory user account</td></tr><tr><td>1</td><td>Group</td><td>Active Directory group.</td></tr><tr><td>2</td><td>Computer</td><td>Active Directory machine account</td></tr><tr><td>3</td><td>OAuthOid</td><td>An open authorization claim of a type not covered by client, role or subject</td></tr><tr><td>4</td><td>OAuthRole</td><td>An open authorization group claim</td></tr><tr><td>5</td><td>OAuthSub-ject</td><td>An open authorization user claim</td></tr><tr><td>6</td><td>OAuthCli-entId</td><td>An open authorization client applic-ation claim</td></tr></table></td></tr><tr><td>ClaimValu-e</td><td>Body</td><td>A string containing the identifying information for the entity specified in the claim. For Active</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>In</th><th>Description</th></tr><tr><td></td><td></td><td>Directory users and groups, this will be in the form DOMAIN\user or group name (e.g. KEYEXAMPLE\PKI Administrators). For Active Directory computers, this will be in the form of a machine account (e.g. KEYEXAMPLE\MyServer$).</td></tr><tr><td>Provider</td><td>Body</td><td>An object containing information about the provider assigned to the security claim.<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID for the provider.</td></tr><tr><td>AuthenticationScheme</td><td>A string indicating the provider auth scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).</td></tr><tr><td>DisplayName</td><td>A string containing the short reference name for the provider (e.g. Active Directory).</td></tr></table></td></tr></table>

Version 1

Version 1 of the PUT /Security/Roles method includes the same capabilities as version 2, but offers support for managing legacy formatted Active Directory identities only.

Important: This has been deprecated since it supports Active Directory identities only. It is retained for backwards compatibility, but all new development should use methods that provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. These newer methods support both Active Directory and other identity providers. See version 2 of this method.

Table 632: PUT Security Roles v1 Input Parameters

Name	In	Description
Id	Body	**Required**. An integer containing the Keyfactor Command identifier for the security role. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
Name	Body	**Required**. A string containing the short reference name for the security role.
Description	Body	**Required**. A string containing the description for the security role.
Enabled	Body	A Boolean that indicates whether the security role is enabled (true) or not (false). Security roles that have been disabled cannot be assigned to security identities. The default is *true*. This is considered deprecated and may be removed in a future release.
Private	Body	A Boolean that indicates whether the security role has been marked private (true) or not (false). The default is *false*. This is considered deprecated and may be removed in a future release.
PermissionSetId	Body	A string indicating the Keyfactor Command reference GUID of the permission set associated with the role (see Permission Sets on page 1959).
Permissions	Body	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Version One Permission Model on page 670 for an overview of the possible permissions. For example: `"Permissions": [ "AdminPortal:Read", "Dashboard:Read" ],`
Identities	Body	An array of objects containing one or more identifiers for each security identity to associate with the role. Supported identifiers include: **Name** / **Description** AccountName — **Required***. A string containing the account name for the security identity. For Active Directory user and groups, this will be in the

Name	In	Description

		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>form DOMAIN\\user or group name. For example:
		KEYEXAMPLE\\PKI Administrators
		
		* One of *AccountName* or *SID* is required in order to specify an identity, but not both.</td></tr><tr><td>SID</td><td>**Required***. A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.
		* One of *AccountName* or *SID* is required in order to specify an identity, but not both.</td></tr></table>
		For example:
		<pre>"Identities": [
		  {
		  "AccountName": "KEYEXAMPLE\\jsmith"
		  },
		  {
		  "AccountName": "KEYEXAMPLE\\mjones"
		  }
		]</pre>

Table 633: PUT Security Roles v1 Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security role.
Name	A string containing the short reference name for the security role.
Description	A string containing the description for the security role.
Enabled	A Boolean that indicates whether the security role is enabled (true) or not (false). Security roles that have been disabled cannot be assigned to security identities. The default is *true*. This is considered deprecated and may be removed in a future release.
Immutable	A Boolean that indicates whether the security role has been marked as editable (false) or not (true). Internal Keyfactor Command roles are not editable. This setting is reserved for Keyfactor Command internal use.
Valid	A Boolean that indicates whether the security role's audit XML is valid (true) or not (false). A security role may become invalid if Keyfactor Command determines that it appears to have been tampered with. This setting is not end-user configurable.
Private	A Boolean that indicates whether the security role has been marked private (true) or not (false). The default is *false*. This is considered deprecated and may be removed in a future release.
PermissionSetId	A string indicating the Keyfactor Command reference GUID of the permission set associated with the role (see Permission Sets on page 1959).
Identities	An array of objects containing information about the security identities assigned to the security role. Identity details include:

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security identity.
AccountName	A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example: `KEYEXAMPLE\\PKI Administrators`
IdentityType	A string indicating the type of identity—User or Group.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SID</td><td>A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.</td></tr>
</table>
</td></tr>
<tr><td>Permissions</td><td>An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See <a>Version One Permission Model on page 670 for an overview of the possible permissions.
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.33.6 POST Security Roles ID Copy

The POST /Security/Roles{id}/Copy method is used to copy an existing security role in Keyfactor Command to create a new security role. This method returns HTTP 200 OK on a success with the details of the new security role.

Tip: The following permissions (see <a>Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see <a>Permission Sets on page 1959).

Important: This has been deprecated since it supports Active Directory identities only. It is retained for backwards compatibility, but all new development should use methods that provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. These newer methods support both Active Directory and other identity providers. There is not an equivalent function to this among the newer methods.

⚠ Instead, use the v2 versions of GET /Security/Roles/{id} and POST /Security/Roles (see GET Security Roles ID on page 2082 and POST Security Roles on page 2093).

Table 634: POST Security Roles {id} Copy Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the security role from which to copy role information. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
role	Body	An array containing information about the new security role to create. Role details include: **Name** — **Required**. A string containing the short reference name for the security role. **Description** — **Required**. A string containing the description for the security role.

Table 635: POST Security Roles {id} Copy Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security role.
Name	A string containing the short reference name for the security role.
Description	A string containing the description for the security role.
Enabled	A Boolean that indicates whether the security role is enabled (true) or not (false). Security roles that have been disabled cannot be assigned to security identities. The default is *true*. This is considered deprecated and may be removed in a future release.
Immutable	A Boolean that indicates whether the security role has been marked as editable (false) or not (true). Internal Keyfactor Command roles are not editable. This setting is reserved for Keyfactor Command internal use.
Valid	A Boolean that indicates whether the security role's audit XML is valid (true) or not (false). A security role may become invalid if Keyfactor Command determines that it appears to have been tampered with. This setting is not end-user configurable.
Private	A Boolean that indicates whether the security role has been marked private (true) or not (false). The default is *false*. This is considered deprecated and may be removed in a future release.
PermissionSetId	A string indicating the Keyfactor Command reference GUID of the permission set associated with the role (see Permission Sets on page 1959).
Identities	An array of objects containing information about the security identities assigned to the security role. Identity details include:

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security identity.
AccountName	A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example: `KEYEXAMPLE\\PKI Administrators`
IdentityType	A string indicating the type of identity—User or Group.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SID</td><td>A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.</td></tr>
</table>
</td></tr>
<tr><td>Permissions</td><td>An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See <a>Version One Permission Model on page 670 for an overview of the possible permissions.
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.33.7 PUT Security Roles ID Identities

The PUT /Security/Roles{id}/Identities method is used to update security identities assigned to a security role in Keyfactor Command. This method returns HTTP 200 OK on a success with the details of the security identities actively assigned to the security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/modify/

The user must hold a role containing this permission in the *Global Permission Set* (see Permission Sets on page 1959).

Important: This has been deprecated since it supports Active Directory identities only. It is retained for backwards compatibility, but all new development should use methods that provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. These newer methods support both Active Directory and other

identity providers. See the v2 version of PUT /Security/Roles to update claims on a security role (see PUT Security Roles on page 2105).

Table 636: PUT Security Roles {id} Identities Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the security role for which to update identities. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.
identities	Body	An array in which you provide a **complete** list of the identities that are associated with an Security Role Id. Use the *GET /Security/Identities* method (see GET Security Identities on page 2034) to retrieve a list of all the security identities to determine the identity ID(s).

Table 637: PUT Security Roles {id} Identities Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security identity.
Name	A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example: `KEYEXAMPLE\\PKI Administrators`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.33.8 GET Security Roles ID Identities

The GET /Security/Roles/{id}/Identities method is used to return the security identities assigned to a security role by security role ID. This method returns HTTP 200 OK on a success with details of the security identities assigned to the specified security role.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/security/read/

Important: This has been deprecated since it supports Active Directory identities only. It is retained for backwards compatibility, but all new development should use methods that provide support for alternate identity providers and the newer claims-based authentication model that accompanies this. These newer methods support both Active Directory and other identity providers. See the v2 version of GET /Security/Roles/{id} to review claims on a security role (see GET Security Roles ID on page 2082).

Table 638: GET Security Roles {id} Identities Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID of the security role for which to retrieve security identities. Use the *GET /Security/Roles* method (see GET Security Roles on page 2088) to retrieve a list of all the security roles to determine the role's ID.

Table 639: GET Security Roles {id} Identities Response Data

Name	Description
Id	An integer containing the Keyfactor Command identifier for the security identity.
Name	A string containing the account name for the security identity. For Active Directory users and groups, this will be in the form DOMAIN\\user or group name. For example: `KEYEXAMPLE\\PKI Administrators`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.34 SSH

The SSH component of the Keyfactor API includes methods necessary to create, update, and delete SSH keys, logons, servers, server groups, and service accounts within Keyfactor Command.

Table 640: SSH Endpoints

Endpoint	Method	Description	Link
/Keys/Unmanaged/{id}	DELETE	Delete a discovered unmanaged SSH key for the specified ID.	DELETE SSH Keys Unman-

Endpoint	Method	Description	Link
			aged ID on page 2128
/Keys/Unmanaged/{id}	GET	Retrieve details for a discovered unmanaged SSH key for the specified ID.	GET SSH Keys Unmanaged ID on page 2129
/Keys/MyKey	GET	Retrieve details for a user's SSH key generated through Keyfactor Command.	GET SSH Keys My Key on page 2130
/Keys/MyKey	POST	Generate a new SSH key pair for a user through Keyfactor Command.	POST SSH Keys My Key on page 2133
/Keys/MyKey	PUT	Update an SSH key for a user through Keyfactor Command.	PUT SSH Keys My Key on page 2137
/Keys/Unmanaged	DELETE	Delete one or more discovered unmanaged SSH keys based on a selection query.	DELETE SSH Keys Unmanaged on page 2140
/Keys/Unmanaged	GET	Retrieve details for one or more discovered unmanaged SSH keys based on a selection query.	GET SSH Keys Unmanaged on page 2141
/Logons/{id}	DELETE	Deletes a Linux logon from Keyfactor Command.	DELETE SSH Logons ID on page 2144
/Logons/{id}	GET	Returns information about a Linux logons.	GET SSH Logons ID on page 2145
/Logons/	GET	Returns information about one or more Linux logons.	GET SSH Logons on page 2147
/Logons/	POST	Creates a new Linux logon in Keyfactor Command and, for servers in *inventory and publish policy* mode, publishes it out to a Linux server.	POST SSH Logons on page 2149

Endpoint	Method	Description	Link
/Logons/Access	POST	Maps users and service accounts with a Linux logon to associate the SSH keys of the users with the Linux logon.	POST SSH Logons Access on page 2152
/Servers/{id}	DELETE	Deletes the SSH server with the specified ID.	DELETE SSH Servers ID on page 2154
/Servers/{id}	GET	Returns the SSH server with the specified ID.	GET SSH Servers ID on page 2155
/Servers/Access/{id}	GET	Retrieves Linux logons along with users and service accounts granted access to those logons for the specified SSH server.	GET SSH Servers Access ID on page 2159
/Servers/	GET	Returns a list of a SSH servers configured in Keyfactor Command.	GET SSH Servers on page 2162
/Servers/	POST	Creates a new SSH server.	POST SSH Servers on page 2167
/Servers/	PUT	Updates an existing SSH server.	PUT SSH Servers on page 2172
/Servers/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server.	DELETE SSH Servers Access on page 2177
/Servers/Access	POST	Creates Linux logon to user and service account mappings for an SSH server.	POST SSH Servers Access on page 2181
/ServerGroups/{id}	DELETE	Deletes the SSH server group with the specified ID.	DELETE SSH Server Groups ID on page 2185

Endpoint	Method	Description	Link
/ServerGroups/{id}	GET	Returns the SSH server group with the specified ID.	GET SSH Server Groups ID on page 2185
/ServerGroups/{name}	GET	Returns the SSH server group with the specified name.	GET SSH Server Groups Name on page 2190
/ServerGroups/Access/{id}	GET	Retrieves Linux logons along with users and service accounts granted access to those logons for the specified SSH server group.	GET SSH Server Groups Access ID on page 2194
/ServerGroups/	GET	Returns a list of a SSH server groups configured in Keyfactor Command.	GET SSH Server Groups on page 2197
/ServerGroups/	POST	Creates a new SSH server group.	POST SSH Server Groups on page 2202
/ServerGroups/	PUT	Updates an existing SSH server group.	PUT SSH Server Groups on page 2210
/ServerGroups/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server group.	DELETE SSH Server Groups Access on page 2218
/ServerGroups/Access	POST	Creates Linux logon to user and service account mappings for an SSH server group.	POST SSH Server Groups Access on page 2221
/ServiceAccounts/{id}	DELETE	Deletes the SSH service account with the specified ID.	DELETE SSH Service Accounts ID on page 2224
/ServiceAccounts/{id}	GET	Returns the SSH service account with the specified ID.	GET SSH Service Accounts ID on

Endpoint	Method	Description	Link
			page 2227
/ServiceAccounts/Key/{id}	GET	Returns the public key and optional private key of an SSH service account with the specified ID.	GET SSH Service Accounts Key ID on page 2234
/ServiceAccounts/	DELETE	Deletes one or more SSH service accounts with the specified IDs.	DELETE SSH Service Accounts on page 2238
/ServiceAccounts/	GET	Returns a list of SSH service accounts based on the specified filters.	GET SSH Service Accounts on page 2240
/ServiceAccounts/	POST	Creates a new SSH service account.	POST SSH Service Accounts on page 2248
/ServiceAccounts/	PUT	Updates an existing SSH service account.	PUT SSH Service Accounts on page 2258
/ServiceAccounts/Rotate/{id}	POST	Generates a new key pair for an existing service account.	POST SSH Service Accounts Rotate ID on page 2266
/Users/{id}	DELETE	Deletes the SSH user with the specified ID.	DELETE SSH Users ID on page 2270
/Users/{id}	GET	Returns the SSH user with the specified ID.	GET SSH Users ID on page 2271
/Users/	GET	Returns a list of SSH users based on the specified filters.	GET SSH Users on page 2276

Endpoint	Method	Description	Link
/Users/	POST	Creates a new SSH user.	POST SSH Users on page 2287
/Users/	PUT	Updates an existing SSH user.	PUT SSH Users on page 2289
/Users/Access	POST	Creates a mapping from the SSH user to one or more Linux logons.	POST SSH Users Access on page 2290

3.6.34.1 SSH Keys

The SSH Keys component of the Keyfactor API includes methods necessary to allow a user with the *SSH User* Keyfactor Command role permission (see SSH Permissions on page 597 in the *Keyfactor Command Reference Guide*) to generate an SSH key pair for himself or herself, retrieve that key, update it, or delete it. Methods are also included to list and delete unmanaged keys—keys discovered on servers configured in inventory only mode.

Table 641: SSH Keys Endpoints

Endpoint	Method	Description	Link
/Unmanaged/{id}	DELETE	Delete a discovered unmanaged SSH key for the specified ID.	DELETE SSH Keys Unmanaged ID on the next page
/Unmanaged/{id}	GET	Retrieve details for a discovered unmanaged SSH key for the specified ID.	GET SSH Keys Unmanaged ID on page 2129
/MyKey	GET	Retrieve details for a user's SSH key generated through Keyfactor Command.	GET SSH Keys My Key on page 2130
/MyKey	POST	Generate a new SSH key pair for a user through Keyfactor Command.	POST SSH Keys My Key on page 2133
/MyKey	PUT	Update an SSH key for a user through Keyfactor Command.	PUT SSH Keys My Key on page 2137
Unmanaged	DELETE	Delete one or more discovered unmanaged SSH keys based on a selection query.	DELETE SSH Keys Unmanaged on page 2140
Unmanaged	GET	Retrieve details for one or more	GET SSH Keys Unman-

Endpoint	Method	Description	Link
		discovered unmanaged SSH keys based on a selection query.	aged on page 2141

DELETE SSH Keys Unmanaged ID

The DELETE /SSH/Keys/Unmanaged/{id} method is used to delete an unmanaged SSH key by ID. Keys discovered on SSH servers during inventory and discovery are considered unmanaged. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Note: Deleting an unmanaged key when the associated server is still in inventory only mode will not delete the key on the target server. The next time the server is scanned, the key will re-appear in Keyfactor Command. See Unmanaged SSH Keys on page 556 in the *Keyfactor Command Reference Guide* for more information.

Table 642: DELETE SSH Keys Unmanaged {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the unmanaged SSH key to be deleted. Use the *GET /SSH/Keys/Unmanaged* method (see GET SSH Keys Unmanaged on page 2141) to retrieve a list of all the unmanaged keys to determine the unmanaged key's ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Keys Unmanaged ID

The GET /SSH/Keys/Unmanaged/{id} method is used to retrieve an unmanaged SSH key by ID. Keys discovered on SSH servers during inventory and discovery are considered unmanaged. This method returns HTTP 200 OK on a success with details for the requested SSH key.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 643: GET SSH Keys Unmanaged {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the unmanaged SSH key to be retrieved. Use the *GET /SSH/Keys/Unmanaged* method (see GET SSH Keys Unmanaged on page 2141) to retrieve a list of all the unmanaged keys to determine the unmanaged key's ID.

Table 644: GET SSH Keys Unmanaged {id} Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH key.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
DiscoveredDate	A string indicating the date, in UTC, on which the SSH key was discovered.
Comments	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. A key may appear with more than one comment if the originating authorized_keys file contained more than one comment.
LogonCount	An integer indicating the number of Linux logons associated with the SSH key.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Keys My Key

The GET /SSH/Keys/MyKey method is used to retrieve the current user's SSH key generated in Keyfactor Command (see POST SSH Keys My Key on page 2133). This method returns HTTP 200 OK on a success with the key's details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/user/
OR
/ssh/server_admin/
OR
/ssh/enterprise_admin/

Table 645: GET SSH Keys My Key Input Parameters

Name	In	Description
includePrivateKey	Query	A Boolean that sets whether to include the private key of the SSH key pair in the response (true) or not (false). If set to *true*, the *x-keyfactor-key-passphrase* header must be supplied. The default is *false*.
x-keyfactor-key-pass-phrase	Header	**Required***. A string that sets a password used to secure the private key of the SSH key pair for download. This field is **required** if *IncludePrivateKey* is set to *true*. **Tip:** This password does not need to match the password entered to secure the private key when the SSH key pair was initially generated. The private key is encrypted at download time and a different password may be used for each download.

Table 646: GET SSH Keys My Key Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the user's SSH key pair.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair.
PrivateKey	A string indicating the private key of the key pair.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key pair. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string indicating the date, in UTC, on which the SSH key pair was created.
StaleDate	A string indicating the date, in UTC, on which the SSH key pair will be considered to have reached the end of its lifetime. By default, the lifetime of an SSH key pair is 365 days. The SSH lifetime is defined by the *Key Lifetime (days)* application setting. See in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the user who requested the key. This email address is used to alert the user when the key pair is approaching the end of its lifetime.
Comments	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command My SSH Key portal or with the POST /SSH/Keys/MyKey method will contain only one string in the array.
LogonCount	An integer indicating the number of Linux logons associated with the SSH key pair.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Keys My Key

The POST /SSH/Keys/MyKey method is used to generate a new SSH key pair for the current user in Keyfactor Command. The user needs to download the private key as an encrypted file and store it locally and an administrator needs to use Keyfactor Command to associate the user's Keyfactor user account with his or her Linux logon account(s) on the target server(s) that the user wishes to access via SSH (see POST SSH Logons Access on page 2152, POST SSH Server Groups Access on page 2221, and POST SSH Servers Access on page 2181). This method returns HTTP 200 OK on a success with the key's details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/user/
OR
/ssh/server_admin/
OR
/ssh/enterprise_admin/

Table 647: POST SSH Keys My Key Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>KeyType</td><td>Body</td><td>Required. A string indicating the cryptographic algorithm to use to generate the SSH key. Possible values are:

<table><tr><th>Numeric Value</th><th>Text Value</th></tr><tr><td>1</td><td>ECDSA</td></tr><tr><td>2</td><td>Ed25519</td></tr><tr><td>3</td><td>RSA</td></tr></table>
The KeyType may be specified using either the numeric value or text value.</td></tr>
<tr><td>PrivateKeyFormat</td><td>Body</td><td>Required. A string indicating the format to use for the downloadable private key. Possible values are:

<table><tr><th>Numeric Value</th><th>Text Value</th></tr><tr><td>1</td><td>OpenSSH</td></tr><tr><td>2</td><td>PKCS8</td></tr></table>
The PrivateKeyFormat may be specified using either the numeric value or text value.</td></tr>
<tr><td>KeyLength</td><td>Body</td><td>Required*. An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA. This field is optional if the KeyType is set to ECDSA or Ed25519 and required if the KeyType is set to RSA.</td></tr>
<tr><td>Email</td><td>Body</td><td>Required. A string containing the email address of the user who requested the key. This email address is used to alert the user when the key pair is approaching the end of its lifetime.</td></tr>
<tr><td>Password</td><td>Body</td><td>Required. A string that sets a password used to secure the private key of the SSH key pair for download.

Tip: This password is used to secure the private key in the downloaded copy of the SSH key pair. You may later download the SSH key pair with private key (see <a>GET SSH Keys My Key on page 2130) and encrypt it with a different pass-</td></tr>
</table>

Name	In	Description
		word, if desired.
Comment	Body	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. **Note:** Although this field is actually an array, entry of only a single comment string is supported. The field is defined as an array to support multiple comments on existing SSH keys found on servers during inventory and discovery.

Table 648: POST SSH Keys My Key Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the user's SSH key pair.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair.
PrivateKey	A string indicating the private key of the key pair.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key pair. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string indicating the date, in UTC, on which the SSH key pair was created.
StaleDate	A string indicating the date, in UTC, on which the SSH key pair will be considered to have reached the end of its lifetime. By default, the lifetime of an SSH key pair is 365 days. The SSH lifetime is defined by the *Key Lifetime (days)* application setting. See in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the user who requested the key. This email address is used to alert the user when the key pair is approaching the end of its lifetime.
Comments	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command My SSH Key portal or with the POST /SSH/Keys/MyKey method will contain only one string in the array.
LogonCount	An integer indicating the number of Linux logons associated with the SSH key pair.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT SSH Keys My Key

The PUT /SSH/Keys/MyKey method is used to update the existing SSH key pair for the current user in Keyfactor Command. Most features of a key pair are fixed and cannot be changed. Only the email address and comment associated with the key may be changed with this option. This method returns HTTP 200 OK on a success with the key's details.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/user/
OR
/ssh/server_admin/
OR
/ssh/enterprise_admin/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 649: PUT SSH Keys My Key Input Parameters

Name	In	Description
ID	Body	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH key.
Email	Body	**Required**. A string containing the email address of the user who requested the key. This email address is used to alert the user when the key pair is approaching the end of its lifetime.
Comment	Body	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. **Note:** Although this field is actually an array, entry of only a single comment string is supported. The field is defined as an array to support multiple comments on existing SSH keys found on servers during inventory and discovery.

Table 650: PUT SSH Keys My Key Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the user's SSH key pair.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key pair. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string indicating the date, in UTC, on which the SSH key pair was created.
StaleDate	A string indicating the date, in UTC, on which the SSH key pair will be considered to have reached the end of its lifetime. By default, the lifetime of an SSH key pair is 365 days. The SSH lifetime is defined by the *Key Lifetime (days)* application setting. See Application Settings: SSH Tab on page 620 in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the user who requested the key. This email address is used to alert the user when the key pair is approaching the end of its lifetime.
Comments	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command My SSH Key portal or with the POST /SSH/Keys/MyKey method will contain only one string in the array.
LogonCount	An integer indicating the number of Linux logons associated with the SSH key pair.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

DELETE SSH Keys Unmanaged

The DELETE /SSH/Keys/Unmanaged method is used to delete one or more unmanaged SSH keys. Keys discovered on SSH servers during inventory and discovery are considered unmanaged. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Note: Deleting an unmanaged key when the associated server is still in inventory only mode will not delete the key on the target server. The next time the server is scanned, the key will re-appear in Keyfactor Command. See Unmanaged SSH Keys on page 556 in the *Keyfactor Command Reference Guide* for more information.

Table 651: DELETE SSH Keys Unmanaged Input Parameters

Name	In	Description
ids	Body	**Required**. An array of integers indicating the Keyfactor Command reference IDs for the unmanaged SSH keys to be deleted provided in the request body in the following format (without parameter name): `[4,27,89]` Use the *GET /SSH/Keys/Unmanaged* method (see GET SSH Keys Unmanaged on the next page) to retrieve a list of all the unmanaged keys to determine the unmanaged key IDs.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Keys Unmanaged

The GET /SSH/Keys/Unmanaged method is used to retrieve one or more unmanaged SSH keys. Keys discovered on SSH servers during inventory and discovery are considered unmanaged. Results can be limited to selected keys using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details for the requested SSH keys.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 652: GET SSH Keys Unmanaged Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Unmanaged Keys Search on page 558. The query fields supported for this endpoint are: • DiscoveredDate • KeyComments • KeyLength • KeyType • ServerId
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 653: GET SSH Keys Unmanaged Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH key.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
DiscoveredDate	A string indicating the date, in UTC, on which the SSH key was discovered.
Comments	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. A key may appear with more than one comment if the originating authorized_keys file contained more than one comment.
LogonCount	An integer indicating the number of Linux logons associated with the SSH key.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.34.2 SSH Logons

The SSH Logons component of the Keyfactor API includes methods necessary to view and manage the Linux user accounts associated with authorized_keys files containing valid SSH public keys. The logons include both those discovered on SSH servers during the initial discovery phase using the orchestrator and those created in Keyfactor Command and published to the SSH servers using the orchestrator.

Table 654: SSH Logon Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes a Linux logon from Keyfactor Command.	DELETE SSH Logons ID below
/{id}	GET	Returns information about a Linux logons.	GET SSH Logons ID on the next page
/	GET	Returns information about one or more Linux logons.	GET SSH Logons on page 2147
/	POST	Creates a new Linux logon in Keyfactor Command and, for servers in *inventory and publish policy* mode, publishes it out to a Linux server.	POST SSH Logons on page 2149
/Access	POST	Maps users and service accounts with a Linux logon to associate the SSH keys of the users with the Linux logon.	POST SSH Logons Access on page 2152

DELETE SSH Logons ID

The DELETE /SSH/Logons/{id} method is used to delete a Linux logon in Keyfactor Command. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server on which the logon exists belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Note: Deleting a logon in Keyfactor Command does not delete it on the Linux server. It must be manually removed from the Linux server at the same time. If this is not done, when the next inventory of the Linux server is performed, the logon will be recreated in Keyfactor Command. This method is intended primarily to be used to clean up logons in Keyfactor Command from SSH servers that have been retired.

Table 655: DELETE SSH Logons {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH logon to be deleted. Use the *GET /SSH/Logons* method (see GET SSH Logons on page 2147) to retrieve a list of all the SSH logons to determine the logon's ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Logons ID

The GET /SSH/Logons/{id} method is used to retrieve a Linux logon by ID. This method returns HTTP 200 OK on a success with details for the requested SSH logon.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server on which the logon exists belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 656: GET SSH Logons {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH logon to retrieve. Use the *GET /SSH/Logons* method (see GET SSH Logons on page 2147) to retrieve a list of all the SSH logons to determine the logon's ID.

Table 657: GET SSH Keys Unmanaged {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</td><td>An integer indicating the Keyfactor Command reference ID for the SSH logon.</td></tr>
<tr><td>Username</td><td>A string indicating the user's logon name on the Linux server.</td></tr>
<tr><td>Server</td><td>An object containing details about the server on which the SSH logon resides. Server information includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the server on which the SSH logon resides.</td></tr>
<tr><td>Hostname</td><td>A string indicating the hostname of the SSH server on which the SSH logon resides. See <a>SSH Servers on page 577 for more information.</td></tr>
<tr><td>UnderManagement</td><td>A Boolean indicating whether the server on which the SSH logon resides is in inventory only mode (false) or inventory and publish policy mode (true).</td></tr>
<tr><td>GroupName</td><td>A string indicating the server group to which the server referenced by Hostname belongs. See <a>SSH Server Groups on page 560 for more information.</td></tr>
</table></td></tr>
<tr><td>KeyCount</td><td>An integer indicating the number of SSH keys associated with the Linux logons.</td></tr>
<tr><td>Access</td><td>An array of objects providing information about the users mapped to the logon. Access information includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of a user or service account that has been associated with the logon. See <a>SSH on page 525 for more information.</td></tr>
<tr><td>Username</td><td>A string indicating the username of a user (in DOMAIN\\username format) or service account (in username@hostname format) that has been associated with the logon.</td></tr>
</table></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Logons

The GET /SSH/Logons method is used to retrieve one or more Linux logons. Results can be limited to selected logons using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details for the requested SSH logons.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server on which the logon exists belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 658: GET SSH Logons Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Logons Search on page 590. The query fields supported for this endpoint are: • Id (Login ID) • LastLogon • Hostname (Logon Server Name) • LogonUserUsername • ServerId • UnmanagedKeyId • Username
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Username*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 659: GET SSH Logons Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH logon.
Username	A string indicating the user's logon name on the Linux server.
ServerId	An integer indicating the Keyfactor Command reference ID of the server on which the SSH logon resides.
ServerName	A string indicating the hostname of the SSH server on which the SSH logon resides. See SSH Servers on page 577 for more information.
GroupName	A string indicating the server group to which the server referenced by *ServerName* belongs. See SSH Server Groups on page 560 for more information.
ServerUnderManagement	A Boolean indicating whether the server on which the SSH logon resides is in *inventory only* mode (false) or *inventory and publish policy* mode (true).
KeyCount	An integer indicating the number of SSH keys associated with the Linux logons.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Logons

The POST /SSH/Logons method is used to create a new Linux logon in Keyfactor Command and, for servers in *inventory and publish policy* mode, publish it out to a Linux server. The logon can optionally be associated with one or more SSH keys by mapping the logon to one or more *users* or *service accounts* during creation. This method returns HTTP 200 OK on a success with details for the new SSH logon.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/

SSH actions are affected by ownership on the server group to which the server on which the logon exists belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 660: POST SSH Logons Input Parameters

Name	In	Description
Username	Body	**Required**. A string indicating the user's logon name on the Linux server.
ServerId	Body	**Required**. An integer indicating the Keyfactor Command reference ID of the server on which the SSH logon should be created. Use the *GET /SSH/Servers* method (see GET SSH Servers on page 2162) to retrieve a list of all the SSH servers to determine the server's ID.
UserIds	Body	An array of integers indicating the Keyfactor Command reference IDs for the users and/or service accounts with which the logon should be associated, provided in the following format: `[4,7,19]` See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information about users and service accounts. Use the *GET /SSH/Users* method (see GET SSH Users on page 2276) to retrieve a list of all the users (including service accounts) created in Keyfactor Command to determine a user's ID.

Table 661: POST SSH Logons Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</td><td>An integer indicating the Keyfactor Command reference ID for the SSH logon.</td></tr>
<tr><td>Username</td><td>A string indicating the user's logon name on the Linux server.</td></tr>
<tr><td>Server</td><td>An object containing details about the server on which the SSH logon resides. Server information includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the server on which the SSH logon resides.</td></tr>
<tr><td>Hostname</td><td>A string indicating the hostname of the SSH server on which the SSH logon resides. See <a>SSH Servers on page 577 for more information.</td></tr>
<tr><td>UnderManagement</td><td>A Boolean indicating whether the server on which the SSH logon resides is in inventory only mode (false) or inventory and publish policy mode (true).</td></tr>
<tr><td>GroupName</td><td>A string indicating the server group to which the server referenced by Hostname belongs. See <a>SSH Server Groups on page 560 for more information.</td></tr>
</table></td></tr>
<tr><td>KeyCount</td><td>An integer indicating the number of SSH keys associated with the Linux logons.</td></tr>
<tr><td>Access</td><td>An array of objects providing information about the users mapped to the logon. Access information includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of a user or service account that has been associated with the logon. See <a>SSH on page 525 for more information.</td></tr>
<tr><td>Username</td><td>A string indicating the username of a user (in DOMAIN\\username format) or service account (in username@hostname format) that has been associated with the logon.</td></tr>
</table></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Logons Access

The POST /SSH/Logons/Access method is used to associate one or more SSH keys with a Linux logon by mapping the logon to one or more *users* or *service accounts*. This method returns HTTP 200 OK on a success with a list of the users associated with the logon.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server on which the logon exists belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 662: POST SSH Logons Access Input Parameters

Name	In	Description
LogonId	Body	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH logon. Use the *GET /SSH/Logons* method (see GET SSH Logons on page 2147) to retrieve a list of all the SSH logons to determine the logon's ID.
UserIds	Body	An array of integers indicating the Keyfactor Command reference IDs for the users and/or service accounts with which the logon should be associated, provided in the following format: `[4,7,19]` Use the *GET /SSH/Users* method (see GET SSH Users on page 2276) to retrieve a list of all the users (including service accounts) created in Keyfactor Command to determine a user's ID. See SSH on page 525 for more information about users and service accounts.

Table 663: POST SSH Logons Access Response Data

Name	Description
LogonId	An integer indicating the Keyfactor Command reference ID for the SSH logon.
LogonName	A string indicating the user's logon name on the Linux server.
Users	An array of objects providing information about the users mapped to the logon. User information includes: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of a *user* or *service account* that has been associated with the logon. See SSH on page 525 for more information. Username — A string indicating the username of a *user* (in DOMAIN\\username format) or *service account* (in username@hostname format) that has been associated with the logon.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.34.3 SSH Servers

The SSH Servers component of the Keyfactor API includes methods necessary to create, update, and delete SSH servers within Keyfactor Command.

Table 664: SSH Servers Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the SSH server with the specified ID.	DELETE SSH Servers ID on the next page
/{id}	GET	Returns the SSH server with the specified ID.	GET SSH Servers ID on page 2155
/Access/{id}	GET	Retrieves Linux logons along with users and	GET SSH

Endpoint	Method	Description	Link
		service accounts granted access to those logons for the specified SSH server.	Servers Access ID on page 2159
/	GET	Returns a list of a SSH servers configured in Keyfactor Command.	GET SSH Servers on page 2162
/	POST	Creates a new SSH server.	POST SSH Servers on page 2167
/	PUT	Updates an existing SSH server.	PUT SSH Servers on page 2172
/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server.	DELETE SSH Servers Access on page 2177
/Access	POST	Creates Linux logon to user and service account mappings for an SSH server.	POST SSH Servers Access on page 2181

DELETE SSH Servers ID

The DELETE /SSH/Servers/{id} method is used to delete an SSH server in Keyfactor Command. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 665: DELETE SSH Servers {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH server to be deleted. Use the *GET /SSH/Servers* method (see GET SSH Servers on page 2162) to retrieve a list of all the SSH servers to determine the server's ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Servers ID

The GET /SSH/Servers/{id} method is used to retrieve an SSH server with the specified ID from Keyfactor Command. This method returns HTTP 200 OK on a success with details for the specified SSH server.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 666: GET SSH Servers {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH server to be retrieved. Use the *GET /SSH/Servers* method (see GET SSH Servers on page 2162) to retrieve a list of all the SSH servers to determine the server's ID.

Table 667: GET SSH Servers {id} Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH server. This ID is automatically set by Keyfactor Command.
AgentId	A string indicating the Keyfactor Command reference GUID for the SSH orchestrator controlling the SSH server.
Hostname	A string indicating the hostname of the SSH server.
ServerGroupId	A string indicating the Keyfactor Command reference GUID for the SSH server group to which the server belongs.
SyncSchedule	An object providing the inventory schedule for the SSH server group to which the SSH server belongs. Inventory schedules cannot be set on an individual SSH server basis. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
For example:
<pre>"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"
      ],
      "Time": "2022-11-20T14:00:00Z"
   }
}</pre>
</td></tr>
<tr><td>Under-Management</td><td>A Boolean indicating whether the SSH server is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Tip: If the server group associated with the SSH server is in inventory and publish policy mode, you will not be able to configure the server in inventory only mode.</td></tr>
<tr><td>Owner</td><td>An object that indicates the Active Directory user who owns the server group to which the server belongs. The owner can only be set by a Keyfactor Command user with the SSH Enterprise Admin role. See SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information. Owner parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the user who holds the owner role on the SSH server group to which the SSH server belongs.</td></tr>
<tr><td>Username</td><td>A string indicating the username of the user (in DOMAIN\\username format) who holds the owner role on the SSH server group to which the SSH server belongs.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>A string indicating the SSH server group to which the SSH server belongs. See SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Orchestrator</td><td>A string indicating the name the SSH orchestrator provided to Keyfactor Command when it registered. This value is configurable when the orchestrator is installed.
For more information about the orchestrator, see Bash Orchestrator on page 2991 in the Keyfactor Orchestrators Installation and Configuration Guide.</td></tr>
<tr><td>Port</td><td>An integer indicating the port that is configured for SSH on the SSH server. The default is 22.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Servers Access ID

The GET /SSH/Servers/Access/{id} method is used to retrieve Linux logons for an SSH server, along with any users or service accounts mapped to those logons, from Keyfactor Command for the

specified server ID. This method returns HTTP 200 OK on a success with details of the logons and associated users, if applicable, for the specified SSH server.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 668: GET SSH Servers Access {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH server for which to retrieve logon and user mappings. Use the *GET /SSH/Servers* method (see GET SSH Servers on page 2162) to retrieve a list of all the SSH servers to determine the server's ID.

Table 669: GET SSH Servers Access {id} Response Data

Name	Description
ServerId	An integer indicating the Keyfactor Command reference ID for the SSH server.
LogonUsers	An array of objects containing information for the Linux logons from the Linux server that have been stored in Keyfactor Command. Possible information includes: **Name** / **Description** LogonId — An integer indicating the Keyfactor Command reference ID of the Linux logon. LogonName — A string indicating the name of the Linux logon. Users — An array of objects containing information about the users and/or service accounts defined in Keyfactor Command that have been mapped to the Linux logon. User information includes: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of a *user* or *service account* that has been associated with the logon. See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information. Username — A string indicating the username of a *user* (in DOMAIN\\username format) or *service account* (in username@hostname format) that has been associated with the logon.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Servers

The GET /SSH/Servers method is used to retrieve one or more SSH servers defined in Keyfactor Command. Results can be limited to selected servers using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details for the requested SSH servers.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 670: GET SSH Servers Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the SSH Server Search on page 583. The query fields supported for this endpoint are: • Agent (Agent ID) • Hostname • Orchestrator (ClientMachine) • ServerGroup (Server Group Id) • ServerGroupName • ServerGroupOwner (Username) • EnforcePublishPolicy (UnderManagement) (true, false)
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Hostname*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 671: GET SSH Servers Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH server. This ID is automatically set by Keyfactor Command.
AgentId	A string indicating the Keyfactor Command reference GUID for the SSH orchestrator controlling the SSH server.
Hostname	A string indicating the hostname of the SSH server.
ServerGroupId	A string indicating the Keyfactor Command reference GUID for the SSH server group to which the server belongs.
SyncSchedule	An object providing the inventory schedule for the SSH server group to which the SSH server belongs. Inventory schedules cannot be set on an individual SSH server basis. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
For example:
<pre>"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"
      ],
      "Time": "2022-11-20T14:00:00Z"
   }
}</pre>
</td></tr>
<tr><td>Under-Management</td><td>A Boolean indicating whether the SSH server is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Tip: If the server group associated with the SSH server is in inventory and publish policy mode, you will not be able to configure the server in inventory only mode.</td></tr>
<tr><td>Owner</td><td>An object that indicates the Active Directory user who owns the server group to which the server belongs. The owner can only be set by a Keyfactor Command user with the SSH Enterprise Admin role. See <a>SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information. Owner parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the user who holds the owner role on the SSH server group to which the SSH server belongs.</td></tr>
<tr><td>Username</td><td>A string indicating the username of the user (in DOMAIN\\username format) who holds the owner role on the SSH server group to which the SSH server belongs.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>A string indicating the SSH server group to which the SSH server belongs. See <a>SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Orchestrator</td><td>A string indicating the name the SSH orchestrator provided to Keyfactor Command when it registered. This value is configurable when the orchestrator is installed.
For more information about the orchestrator, see <a>Bash Orchestrator on page 2991 in the Keyfactor Orchestrators Installation and Configuration Guide.</td></tr>
<tr><td>Port</td><td>An integer indicating the port that is configured for SSH on the SSH server. The default is 22.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Servers

The POST /SSH/Servers method is used to create a new SSH server in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the new SSH server.

Before adding a new SSH server, be sure that you have added at least one server group (see POST SSH Server Groups on page 2202) and that your Keyfactor Bash Orchestrator has been registered and approved in Keyfactor Command (see GET Agents on page 858).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 672: POST SSH Servers Input Parameters

Name	In	Description
AgentId	Body	**Required**. A string indicating the Keyfactor Command reference GUID for the SSH orchestrator controlling the SSH server.
Hostname	Body	**Required**. A string indicating the hostname of the SSH server.
ServerGroupId	Body	**Required**. A string indicating the Keyfactor Command reference GUID for the SSH server group to which the server belongs.
UnderManagement	Body	A Boolean indicating whether the SSH server is in *inventory only* mode (False) or *inventory and publish policy* mode (True). **Tip:** If the server group associated with the SSH server is in *inventory and publish policy* mode, you will not be able to configure the server in *inventory only* mode.
Port	Body	An integer indicating the port that is configured for SSH on the SSH server. The default is 22.

Table 673: POST SSH Servers Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH server. This ID is automatically set by Keyfactor Command.
AgentId	A string indicating the Keyfactor Command reference GUID for the SSH orchestrator controlling the SSH server.
Hostname	A string indicating the hostname of the SSH server.
ServerGroupId	A string indicating the Keyfactor Command reference GUID for the SSH server group to which the server belongs.
SyncSchedule	An object providing the inventory schedule for the SSH server group to which the SSH server belongs. Inventory schedules cannot be set on an individual SSH server basis. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
For example:
<pre>"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"
      ],
      "Time": "2022-11-20T14:00:00Z"
   }
}</pre>
</td></tr>
<tr><td>Under-Management</td><td>A Boolean indicating whether the SSH server is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Tip: If the server group associated with the SSH server is in inventory and publish policy mode, you will not be able to configure the server in inventory only mode.</td></tr>
<tr><td>Owner</td><td>An object that indicates the Active Directory user who owns the server group to which the server belongs. The owner can only be set by a Keyfactor Command user with the SSH Enterprise Admin role. See <a>SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information. Owner parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the user who holds the owner role on the SSH server group to which the SSH server belongs.</td></tr>
<tr><td>Username</td><td>A string indicating the username of the user (in DOMAIN\\username format) who holds the owner role on the SSH server group to which the SSH server belongs.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>A string indicating the SSH server group to which the SSH server belongs. See <a>SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Orchestrator</td><td>A string indicating the name the SSH orchestrator provided to Keyfactor Command when it registered. This value is configurable when the orchestrator is installed.
For more information about the orchestrator, see <a>Bash Orchestrator on page 2991 in the Keyfactor Orchestrators Installation and Configuration Guide.</td></tr>
<tr><td>Port</td><td>An integer indicating the port that is configured for SSH on the SSH server. The default is 22.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT SSH Servers

The PUT /SSH/Servers method is used to update an existing SSH server in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the SSH server.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 674: PUT SSH Servers Input Parameters

Name	In	Description
ID	Body	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH server. This ID is automatically set by Keyfactor Command.
UnderManagement	Body	A Boolean indicating whether the SSH server is in *inventory only* mode (False) or *inventory and publish policy* mode (True). **Tip:** If the server group associated with the SSH server is in *inventory and publish policy* mode, you will not be able to configure the server in *inventory only* mode.
Port	Body	An integer indicating the port that is configured for SSH on the SSH server. The default is 22.

Table 675: PUT SSH Servers Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH server. This ID is automatically set by Keyfactor Command.
AgentId	A string indicating the Keyfactor Command reference GUID for the SSH orchestrator controlling the SSH server.
Hostname	A string indicating the hostname of the SSH server.
ServerGroupId	A string indicating the Keyfactor Command reference GUID for the SSH server group to which the server belongs.
SyncSchedule	An object providing the inventory schedule for the SSH server group to which the SSH server belongs. Inventory schedules cannot be set on an individual SSH server basis. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description
Off	Turn off a previously configured schedule.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

For example, every hour:

```
"Interval": {
   "Minutes": 60
}
```

Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
For example:
<pre>"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"
      ],
      "Time": "2022-11-20T14:00:00Z"
   }
}</pre>
</td></tr>
<tr><td>Under-Management</td><td>A Boolean indicating whether the SSH server is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Tip: If the server group associated with the SSH server is in inventory and publish policy mode, you will not be able to configure the server in inventory only mode.</td></tr>
<tr><td>Owner</td><td>An object that indicates the Active Directory user who owns the server group to which the server belongs. The owner can only be set by a Keyfactor Command user with the SSH Enterprise Admin role. See SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information. Owner parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the user who holds the owner role on the SSH server group to which the SSH server belongs.</td></tr>
<tr><td>Username</td><td>A string indicating the username of the user (in DOMAIN\\username format) who holds the owner role on the SSH server group to which the SSH server belongs.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>A string indicating the SSH server group to which the SSH server belongs. See SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Orchestrator</td><td>A string indicating the name the SSH orchestrator provided to Keyfactor Command when it registered. This value is configurable when the orchestrator is installed.
For more information about the orchestrator, see Bash Orchestrator on page 2991 in the Keyfactor Orchestrators Installation and Configuration Guide.</td></tr>
<tr><td>Port</td><td>An integer indicating the port that is configured for SSH on the SSH server. The default is 22.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

DELETE SSH Servers Access

The DELETE /SSH/Servers/Access method is used to remove a mapping of Keyfactor Command users or service accounts to one or more Linux logons on one or more SSH servers. This method

returns HTTP 200 OK on a success with details of the logons and remaining associated users, if applicable, for the specified SSH server(s).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Tip: Before deleting a logon to user mapping, be sure that you have switched the server from which you will removing your mapping (or its server group) to *inventory and publish policy* mode so that the key for the user will be removed from the server. If the server is in *inventory only* mode and you remove a mapping for it in Keyfactor Command, the mapping will be removed in Keyfactor Command only and the key for the user will not be removed from the server.

Table 676: DELETE SSH Servers Access Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ServerId</td><td>Body</td><td>Required. An integer indicating the Keyfactor Command reference ID for the SSH server.</td></tr>
<tr><td>LogonUsers</td><td>Body</td><td>Required. An array of objects containing information for the Linux logon(s) to update. The following information should be included:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>LogonName</td><td>A string indicating the name of the Linux logon.</td></tr><tr><td>Users</td><td>An array of strings indicating the user names of one or more users (in DOMAIN\\username format) or service accounts (in username@hostname format) to be removed from association with the logon.</td></tr></table>For example:
<pre>"LogonUsers": [
   {
      "LogonName": "johns",
      "Users": [
         "KEYEXAMPLE\\jsmith"
      ]
   }
]</pre></td></tr>
</table>

Table 677: DELETE SSH Servers Access Response Data

Name	Description
ServerId	An integer indicating the Keyfactor Command reference ID for the SSH server.
LogonUsers	An array of objects containing information for the Linux logons from the Linux server that have been stored in Keyfactor Command. Possible information includes: **Name** / **Description** LogonId — An integer indicating the Keyfactor Command reference ID of the Linux logon. LogonName — A string indicating the name of the Linux logon. Users — An array of objects containing information about the users and/or service accounts defined in Keyfactor Command that have been mapped to the Linux logon. User information includes: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of a *user* or *service account* that has been associated with the logon. See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information. Username — A string indicating the username of a *user* (in DOMAIN\\username format) or *service account* (in username@hostname format) that has been associated with the logon.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Servers Access

The POST /SSH/Servers/Access method is used to create a mapping of one or more Linux logons to Keyfactor Command users or service accounts for one or more SSH servers. This method returns HTTP 200 OK on a success with details of the logons and associated users, if applicable, for the specified SSH server(s).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Tip: Before creating a logon to user mapping, be sure that you have switched the server to which you will add your mapping (or its server group) to *inventory and publish policy* mode so that the key for the user will be published to the server. If the server is in *inventory only* mode and you add a mapping for it in Keyfactor Command, the mapping will appear in Keyfactor Command only and the key for the user will not be published out to the server.

Table 678: POST SSH Servers Access Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ServerId</td><td>Body</td><td>Required. An integer indicating the Keyfactor Command reference ID for the SSH server.</td></tr>
<tr><td>LogonUsers</td><td>Body</td><td>Required. An array of objects containing information for the Linux logon(s) to update. The following information should be included:

<table><tr><th>Name</th><th>Description</th></tr><tr><td>LogonName</td><td>A string indicating the name of the Linux logon.</td></tr><tr><td>Users</td><td>An array of strings indicating the user names of one or more users (in DOMAIN\\username format) or service accounts (in username@hostname format) to be associated with the logon.</td></tr></table>
For example:

<pre>"LogonUsers": [
   {
      "LogonName": "johns",
      "Users": [
         "KEYEXAMPLE\\jsmith"
      ]
   }
]</pre></td></tr>
</table>

Table 679: POST SSH Servers Access Response Data

Name	Description
ServerId	An integer indicating the Keyfactor Command reference ID for the SSH server.
LogonUsers	An array of objects containing information for the Linux logons from the Linux server that have been stored in Keyfactor Command. Possible information includes: **Name** / **Description** LogonId — An integer indicating the Keyfactor Command reference ID of the Linux logon. LogonName — A string indicating the name of the Linux logon. Users — An array of objects containing information about the users and/or service accounts defined in Keyfactor Command that have been mapped to the Linux logon. User information includes: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of a *user* or *service account* that has been associated with the logon. See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information. Username — A string indicating the username of a *user* (in DOMAIN\\username format) or *service account* (in username@hostname format) that has been associated with the logon.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.34.4 SSH Server Groups

The SSH Server Groups component of the Keyfactor API includes methods necessary to create, update and delete SSH server groups within Keyfactor Command.

Table 680: SSH Server Groups Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the SSH server group with the specified ID.	DELETE SSH Server Groups ID on the next page
/{id}	GET	Returns the SSH server group with the specified ID.	GET SSH Server Groups ID on the next page
/{name}	GET	Returns the SSH server group with the specified name.	GET SSH Server Groups Name on page 2190
/Access/{id}	GET	Retrieves Linux logons along with users and service accounts granted access to those logons for the specified SSH server group.	GET SSH Server Groups Access ID on page 2194
/	GET	Returns a list of a SSH server groups configured in Keyfactor Command.	GET SSH Server Groups on page 2197
/	POST	Creates a new SSH server group.	POST SSH Server Groups on page 2202
/	PUT	Updates an existing SSH server group.	PUT SSH Server Groups on page 2210
/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server group.	DELETE SSH Server Groups Access on page 2218
/Access	POST	Creates Linux logon to user and service account mappings for an SSH server group.	POST SSH Server Groups Access on page 2221

DELETE SSH Server Groups ID

The DELETE /SSH/ServerGroups/{id} method is used to delete an SSH server group in Keyfactor Command. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/enterprise_admin/

Table 681: DELETE SSH Server Groups {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID for the SSH server group to be deleted. Use the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 2197) to retrieve a list of all the SSH server groups to determine the server group's GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Server Groups ID

The GET /SSH/ServerGroups/{id} method is used to retrieve an SSH server group with the specified GUID from Keyfactor Command. This method returns HTTP 200 OK on a success with details for the specified SSH server group.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 682: GET SSH Server Groups {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID for the SSH server group to be retrieved. Use the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 2197) to retrieve a list of all the SSH server groups to determine the server group's GUID.

Table 683: GET SSH Server Groups {id} Response Data

Name	Description
ID	A string indicating the Keyfactor Command reference GUID for the SSH server group. This GUID is automatically set by Keyfactor Command.
Owner	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information. Owner parameters are: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group. Username — A string indicating the username of the *user* (in DOMAIN\\username format) who holds the owner role on the SSH server group.
GroupName	A string indicating the name of the SSH server group.
SyncSchedule	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are: **Name** / **Description** Off — Turn off a previously configured schedule. Interval — A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour: `"Interval": { "Minutes": 60 }`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>

For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:

```
"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

For example:

```
"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"
      ],
      "Time": "2022-11-20T14:00:00Z"
```

Name	Description

	<pre>}
	}</pre>

Under-Management	A Boolean indicating whether the SSH server group is in *inventory only* mode (False) or *inventory and publish policy* mode (True).
ServerCount	An integer indicating the number of SSH servers that belong to the server group.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Server Groups Name

The GET /SSH/ServerGroups/{name} method is used to retrieve an SSH server group with the specified name from Keyfactor Command. This method returns HTTP 200 OK on a success with details for the specified SSH server group.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 684: GET SSH Server Groups {name} Input Parameters

Name	In	Description
name	Path	**Required**. A string indicating the full name of the SSH server group to be retrieved.

Table 685: GET SSH Server Groups {name} Response Data

Name	In	Description
ID	Body	A string indicating the Keyfactor Command reference GUID for the SSH server group. This GUID is automatically set by Keyfactor Command.
Owner	Body	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information. Owner parameters are: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group. Username — A string indicating the username of the *user* (in DOMAIN\\username format) who holds the owner role on the SSH server group.
GroupName	Body	A string indicating the name of the SSH server group.
SyncSchedule	Body	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are: **Name** / **Description** Off — Turn off a previously configured schedule. Interval — A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

For example:
</td></tr>
</table>

Name	In	Description

		<pre>"SyncSchedule": {
		  "Weekly": {
		  "Days": [
		  "Monday",
		  "Wednesday",
		  "Friday"
		  ],
		  "Time": "2022-11-20T14:00:00Z"
		  }
		}</pre>

Under-Management	Body	A Boolean indicating whether the SSH server group is in *inventory only* mode (False) or *inventory and publish policy* mode (True).
ServerCount	Body	An integer indicating the number of SSH servers that belong to the server group.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Server Groups Access ID

The GET /SSH/ServerGroups/Access/{id} method is used to retrieve Linux logons for an SSH server group, along with any users or service accounts mapped to those logons, from Keyfactor Command for the specified server group GUID. This method returns HTTP 200 OK on a success with details of the logons and associated users, if applicable, for the specified SSH server group.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 686: GET SSH Server Groups Access {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID for the SSH server group for which to retrieve logon and user mappings. Use the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 2197) to retrieve a list of all the SSH server groups to determine the server group's ID.

Table 687: GET SSH Server Groups Access {id} Response Data

Name	Description
ServerGroupId	A string indicating the Keyfactor Command reference GUID for the SSH server group.
LogonUsers	An array of objects containing information for the Linux logons from the Linux server that have been stored in Keyfactor Command. Possible information includes:

Name	Description
LogonName	A string indicating the name of the Linux logon. **Tip:** Logons only appear in the results if they exist with the same spelling on all servers in the server group.
Users	An array of objects containing information about the users and/or service accounts defined in Keyfactor Command that are mapped to the Linux logon. User information includes:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of a *user* or *service account* that is associated with the logon. See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information.
Username	A string indicating the username of a *user* (in DOMAIN\\username format) or *service account* (in user-name@hostname format) that is associated with the logon.

Tip: Users only appear in the results if they have been mapped to the same logon on all servers in the server group.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Server Groups

The GET /SSH/ServerGroups method is used to retrieve one or more SSH server groups defined in Keyfactor Command. Results can be limited to selected server groups using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details for the requested SSH server groups.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 688: GET SSH Server Groups Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Server Group Search on page 575. The query fields supported for this endpoint are: • GroupId • GroupName • Owner (Owner ID) • OwnerName (Username) • EnforcePublishPolicy (Under Management) (true, false)
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *GroupName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 689: GET SSH Server Groups Response Data

Name	Description
ID	A string indicating the Keyfactor Command reference GUID for the SSH server group. This GUID is automatically set by Keyfactor Command.
Owner	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information. Owner parameters are: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group. Username — A string indicating the username of the *user* (in DOMAIN\\username format) who holds the owner role on the SSH server group.
GroupName	A string indicating the name of the SSH server group.
SyncSchedule	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are: **Name** / **Description** Off — Turn off a previously configured schedule. Interval — A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour: `"Interval": { "Minutes": 60 }`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>

For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:

```
"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

For example:

```
"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"
      ],
      "Time": "2022-11-20T14:00:00Z"
```

Name	Description
	<pre> } }</pre>
Under-Management	A Boolean indicating whether the SSH server group is in *inventory only* mode (False) or *inventory and publish policy* mode (True).
ServerCount	An integer indicating the number of SSH servers that belong to the server group.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Server Groups

The POST /SSH/ServerGroups method is used to create an SSH server groups defined in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the new SSH server group.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/enterprise_admin/

Table 690: POST SSH Server Groups Input Parameters

Name	In	Description
OwnerName	Body	**Required**. A string indicating the Active Directory user who owns the server group (in DOMAIN\\username format). The owner can only be set by a Keyfactor Command user with the *SSH Enterprise Admin* role. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information. **Tip:** Notice that the field name and structure returned on a GET is not the same as that used on a POST and PUT for the server group owner.
GroupName	Body	**Required**. A string indicating the name of the SSH server group.
SyncSchedule	Body	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are: **Name** / **Description** Off — Turn off a previously configured schedule. Interval — A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour: `"Interval": { "Minutes": 60 }` Daily — A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre>
</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
</pre></td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>
"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}
</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
For example:
<pre>
"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
</pre>
</td></tr>
</table>

Name	In	Description

		<pre>"Wednesday",
		"Friday"
		],
		"Time": "2022-11-20T14:00:00Z"
		}
		}</pre>
		The default is unset.

Under-Management	Body	A Boolean indicating whether the SSH server group is in *inventory only* mode (False) or *inventory and publish policy* mode (True). The default is False.

Table 691: POST SSH Server Groups Response Data

Name	Description
ID	A string indicating the Keyfactor Command reference GUID for the SSH server group. This GUID is automatically set by Keyfactor Command.
Owner	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information. Owner parameters are: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group. Username — A string indicating the username of the *user* (in DOMAIN\\username format) who holds the owner role on the SSH server group.
GroupName	A string indicating the name of the SSH server group.
SyncSchedule	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are: **Name** / **Description** Off — Turn off a previously configured schedule. Interval — A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour: `"Interval": { "Minutes": 60 }`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>

For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"</pre></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:

```
"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}
```

</td></tr>
</table>

Note: Although the Keyfactor API Reference and Utility—Swagger—*Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

For example:

```
"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"
      ],
      "Time": "2022-11-20T14:00:00Z"
```

Name	Description
	<pre> } }</pre>
Under-Management	A Boolean indicating whether the SSH server group is in *inventory only* mode (False) or *inventory and publish policy* mode (True).
ServerCount	An integer indicating the number of SSH servers that belong to the server group.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT SSH Server Groups

The PUT /SSH/ServerGroups method is used to update an existing SSH server groups defined in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the updated SSH server group.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 692: PUT SSH Server Groups Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ID</td><td>Body</td><td>A string indicating the Keyfactor Command reference GUID for the SSH server group. This GUID is automatically set by Keyfactor Command.</td></tr>
<tr><td>OwnerName</td><td>Body</td><td>Required. A string indicating the Active Directory user who owns the server group (in DOMAIN\\username format). The owner can only be set by a Keyfactor Command user with the SSH Enterprise Admin role. See <a>SSH Server Groups on page 560 in the Keyfactor Command Reference Guide for more information.

Tip: Notice that the field name and structure returned on a GET is not the same as that used on a POST and PUT for the server group owner.</td></tr>
<tr><td>GroupName</td><td>Body</td><td>Required. A string indicating the name of the SSH server group.</td></tr>
<tr><td>SyncSchedule</td><td>Body</td><td>An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre><code>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</code></pre>
</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre><code>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",</code></pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
</pre></td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>
"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}
</pre></td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
For example:
<pre>
"SyncSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
</pre>
</td></tr>
</table>

Name	In	Description

		<pre>"Wednesday",
		"Friday"
		],
		"Time": "2022-11-20T14:00:00Z"
		}
		}</pre>
		The default is unset.

Under-Management	Body	A Boolean indicating whether the SSH server group is in *inventory only* mode (False) or *inventory and publish policy* mode (True). The default is False.

Table 693: PUT SSH Server Groups Response Data

Name	In	Description
ID	Body	A string indicating the Keyfactor Command reference GUID for the SSH server group. This GUID is automatically set by Keyfactor Command.
Owner	Body	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information. Owner parameters are: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group. Username — A string indicating the username of the *user* (in DOMAIN\\username format) who holds the owner role on the SSH server group.
GroupName	Body	A string indicating the name of the SSH server group.
SyncSchedule	Body	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are: **Name** / **Description** Off — Turn off a previously configured schedule. Interval — A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>

For example, on the first of every month at 5:30 pm:

<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
</table>

  Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.

For example:
</td></tr>
</table>

Name	In	Description

		<pre>"SyncSchedule": {
		  "Weekly": {
		  "Days": [
		  "Monday",
		  "Wednesday",
		  "Friday"
		  ],
		  "Time": "2022-11-20T14:00:00Z"
		  }
		}</pre>

Under-Management	Body	A Boolean indicating whether the SSH server group is in *inventory only* mode (False) or *inventory and publish policy* mode (True).
ServerCount	Body	An integer indicating the number of SSH servers that belong to the server group.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

DELETE SSH Server Groups Access

The DELETE /SSH/ServerGroups/Access method is used to remove a mapping of one or more Linux logons to Keyfactor Command users or service accounts for one or more SSH server groups. This method returns HTTP 200 OK on a success with details of the logons and associated users, if applicable, for the specified SSH server group(s).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Tip: Before deleting a logon to user mapping, be sure that you have switched the server group from which you will removing your mapping to *inventory and publish policy* mode so that the key for the user will be removed from the servers in the server group. If the server group

is in *inventory only* mode and you remove a mapping for it in Keyfactor Command, the mapping will be removed in Keyfactor Command only and the key for the user will not be removed from the servers.

Table 694: DELETE SSH Server Groups Access Input Parameters

Name	In	Description
ServerGroupId	Body	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH server group.
LogonUsers	Body	An array of objects containing information for the Linux logon(s) to update. The following information should be included: **Name** — **Description** **LogonName** — A string indicating the name of the Linux logon. **Users** — An array of strings indicating the user names of one or more *users* (in DOMAIN\\username format) or *service accounts* (in user-name@hostname format) to be removed from association with the logon. For example: `"LogonUsers": [ { "LogonName": "johns", "Users": [ "KEYEXAMPLE\\jsmith" ] } ]`

Table 695: DELETE SSH Server Groups Access {id} Response Data

Name	Description
ServerGroupId	A string indicating the Keyfactor Command reference GUID for the SSH server group.
LogonUsers	An array of objects containing information for the Linux logons from the Linux server that have been stored in Keyfactor Command. Possible information includes:

Name	Description
LogonName	A string indicating the name of the Linux logon. **Tip:** Logons only appear in the results if they exist with the same spelling on all servers in the server group.
Users	An array of objects containing information about the users and/or service accounts defined in Keyfactor Command that are mapped to the Linux logon. User information includes:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of a *user* or *service account* that is associated with the logon. See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information.
Username	A string indicating the username of a *user* (in DOMAIN\\username format) or *service account* (in user-name@hostname format) that is associated with the logon.

Tip: Users only appear in the results if they have been mapped to the same logon on all servers in the server group.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Server Groups Access

The POST /SSH/ServerGroups/Access method is used to create a mapping of one or more Linux logons to Keyfactor Command users or service accounts for one or more SSH server groups. This method returns HTTP 200 OK on a success with details of the logons and associated users, if applicable, for the specified SSH server group(s).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Tip: Before creating a logon to user mapping, be sure that you have switched the server group to which you will add your mapping to *inventory and publish policy* mode so that the key for the user will be published to the servers in the group. If the server group is in *inventory only* mode and you add a mapping for it in Keyfactor Command, the mapping will appear in Keyfactor Command only and the key for the user will not be published out to the servers.

Table 696: POST SSH Server Groups Access Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ServerGroupId</td><td>Body</td><td>Required. A string indicating the Keyfactor Command reference GUID for the SSH server group.</td></tr>
<tr><td>LogonUsers</td><td>Body</td><td>Required. An array of objects containing information for the Linux logon (s) to update. The following information should be included:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>LogonName</td><td>A string indicating the name of the Linux logon.</td></tr><tr><td>Users</td><td>An array of strings indicating the user names of one or more users (in DOMAIN\\username format) or service accounts (in user-name@hostname format) to be associated with the logon.</td></tr></table>For example:
<pre>"LogonUsers": [
   {
      "LogonName": "johns",
      "Users": [
         "KEYEXAMPLE\\jsmith"
      ]
   }
]</pre></td></tr>
</table>

Table 697: POST SSH Server Groups Access {id} Response Data

Name	Description
ServerGroupId	A string indicating the Keyfactor Command reference GUID for the SSH server group.
LogonUsers	An array of objects containing information for the Linux logons from the Linux server that have been stored in Keyfactor Command. Possible information includes:

Name	Description
LogonName	A string indicating the name of the Linux logon. **Tip:** Logons only appear in the results if they exist with the same spelling on all servers in the server group.
Users	An array of objects containing information about the users and/or service accounts defined in Keyfactor Command that are mapped to the Linux logon. User information includes:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of a *user* or *service account* that is associated with the logon. See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information.
Username	A string indicating the username of a *user* (in DOMAIN\\username format) or *service account* (in user-name@hostname format) that is associated with the logon.

Tip: Users only appear in the results if they have been mapped to the same logon on all servers in the server group.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.34.5 SSH Service Accounts

The SSH Service Accounts component of the Keyfactor API includes methods necessary to retrieve, create, update, rotate and delete service accounts and associated keys in Keyfactor Command.

Table 698: SSH Service Accounts Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the SSH service account with the specified ID.	DELETE SSH Service Accounts ID below
/{id}	GET	Returns the SSH service account with the specified ID.	GET SSH Service Accounts ID on page 2227
/Key/{id}	GET	Returns the public key and optional private key of an SSH service account with the specified ID.	GET SSH Service Accounts Key ID on page 2234
/	DELETE	Deletes one or more SSH service accounts with the specified IDs.	DELETE SSH Service Accounts on page 2238
/	GET	Returns a list of SSH service accounts based on the specified filters.	GET SSH Service Accounts on page 2240
/	POST	Creates a new SSH service account.	POST SSH Service Accounts on page 2248
/	PUT	Updates an existing SSH service account.	PUT SSH Service Accounts on page 2258
/Rotate/{id}	POST	Generates a new key pair for an existing service account.	POST SSH Service Accounts Rotate ID on page 2266

DELETE SSH Service Accounts ID

The DELETE /SSH/ServiceAccounts/{id} method is used to delete an SSH service account in Keyfactor Command, including its SSH key pair. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 699: DELETE SSH Service Accounts {id} Input Parameters

Name	In	Description

id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH service account to be deleted.
		Use the *GET /SSH/ServiceAccounts* method (see GET SSH Service Accounts on page 2240) to retrieve a list of all the SSH service accounts to determine the service account's ID.
		
		Tip: Be sure to use the ID of the service account itself and not the ID of the service account user or service account's key within the service account. For example, notice the following record returned from a GET /SSH/ServiceAccounts:
		
		{
		"Id": 2,
		"ClientHostname": "appsrvr80.keyexample.com",
		"ServerGroup": {
		"Id": "603d3d4c-89dd-4ab8-92e1-8e83db3d5546",
		"GroupName": "Server Group Two",
		"UnderManagement": false
		},
		"User": {
		"Id": 7,
		"Key": {
		"Id": 36,
		"Fingerprint": "kwuo2k3Ej7wFVMLhI3g+rx-t2qXwGp7qcvzdBjVTDHNg=",
		"PublicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCAln+t [truncated for display]",
		"KeyType": "RSA",
		"KeyLength": 2048,
		"CreationDate": "2023-11-17T17:53:55.68",
		"Email": "pkiadmins@keyexample.com",
		"Comments": [
		"Access App Two"
		],
		"LogonCount": 3
		},
		"Username": "svc_access2@appsrvr80.keyexample.com"
		}
		}
		
		It contains three IDs:
		• ID 2: The service account's ID. Use this one for delete requests.
		• ID 7: The service account user's ID.
		• ID 36: The ID of the service account user's key.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Service Accounts ID

The GET /SSH/ServiceAccounts/{id} method is used to retrieve an SSH service account from Keyfactor Command. This method returns HTTP 200 OK on a success with details for the requested SSH service account and its public key. To return the SSH private key, use the GET /SSH/ServiceAccounts/Key/{id} method (see GET SSH Service Accounts Key ID on page 2234).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 700: GET SSH Service Accounts {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH service account to be retrieved. Use the *GET /SSH/ServiceAccounts* method (see GET SSH Service Accounts on page 2240) to retrieve a list of all the SSH service accounts to determine the service account's ID.

Table 701: GET SSH Service Accounts {id} Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH service account. This ID is automatically set by Keyfactor Command.
ClientHostname	A string indicating the client hostname reference for the service account key. This field is used for reference only and does not need to match an actual client hostname. It is used when building the full user name of the service account key for mapping to Linux logons for publishing to Linux servers (e.g. username@client_hostname). The naming convention is to enter the hostname of the server on which the application that will use the private key resides (e.g. appsrvr12), but you can put anything you like in this field (e.g. cheesetoast).
ServerGroup	An object that indicates the SSH server group for the service account. The server group is used to control who has access in Keyfactor Command to the service account key. It does not limit where the key can be published. See SSH Permissions on page 597 in the *Keyfactor Command Reference Guide* for more information. Server group information includes:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the SSH server group.
Owner	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information.

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group.

GroupName	A string indicating the name of the SSH server group.
SyncSchedule	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr><tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.<table><tr><th>Name</th><th>Description</th></tr><tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr></table>For example, every hour:<pre>"Interval": {
	  "Minutes": 60
	}</pre></td></tr><tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr></table>For example, daily at 11:30 pm:<pre>"Daily": {</pre></td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td></td><td><pre>
   "Time": "2023-11-25T23:30:00Z"
}
</pre></td></tr>
<tr><td>Week-ly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:m-m:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>Under-Management</td><td>A Boolean indicating whether the SSH server group is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
</table>
</td></tr>
<tr><td>User</td><td>An object containing information about the service account user. Service account user details include:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SSH service account user.</td></tr><tr><td>Key</td><td>An object containing information about the key for the service account user. Key details include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SSH service account's key.</td></tr><tr><td>Fingerprint</td><td>A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.</td></tr><tr><td>PublicKey</td><td>A string indicating the public key of the key pair for the SSH service account.</td></tr><tr><td>KeyType</td><td>A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: - RSA - ECDSA - Ed25519 </td></tr><tr><td>KeyLength</td><td>An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.</td></tr><tr><td>CreationDate</td><td>A string indicating the date, in UTC, on which the SSH key pair was created.</td></tr><tr><td>StaleDate</td><td>A string indicating the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by</td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the Key Lifetime (days) application setting. See <a>Application Settings: SSH Tab on page 620 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Email</td><td>A string containing the email address of the administrator or group of administrators responsible for managing the key. This email address is used to alert the administrator or group of administrators when the key pair is approaching the end of its lifetime.</td></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the service account. The username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
<tr><td>LogonIds</td><td>An array of integers indicating the Keyfactor Command reference IDs of Linux logons that are associated with the service account in order to publish the service account's public key to the servers on which the logons are located.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Service Accounts Key ID

The GET /SSH/ServiceAccounts/Key/{id} method is used to retrieve the key information for an SSH service account from Keyfactor Command. This method returns HTTP 200 OK on a success with details for the requested SSH service account key, including optional private key.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 702: GET SSH Service Accounts Key {id} Input Parameters

Name	In	Description

id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH service account key for which to retrieve key information.
		Use the *GET /SSH/ServiceAccounts* method (see GET SSH Service Accounts on page 2240) to retrieve a list of all the SSH service accounts to determine the service account's key ID.
		
		Tip: Be sure to use the ID of the service account's key and not the ID of the service account itself or the service account user. For example, notice the following record returned from a GET /SSH/ServiceAccounts:
		
		{
		"Id": 2,
		"ClientHostname": "appsrvr80.keyexample.com",
		"ServerGroup": {
		"Id": "603d3d4c-89dd-4ab8-92e1-8e83db3d5546",
		"GroupName": "Server Group Two",
		"UnderManagement": false
		},
		"User": {
		"Id": 7,
		"Key": {
		"Id": 36,
		"Fingerprint": "kwuo2k3Ej7wFVMLhI3g+rx-t2qXwGp7qcvzdBjVTDHNg=",
		"PublicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCAln+t [truncated for display]",
		"KeyType": "RSA",
		"KeyLength": 2048,
		"CreationDate": "2023-11-17T17:53:55.68",
		"Email": "pkiadmins@keyexample.com",
		"Comments": [
		"Access App Two"
		],
		"LogonCount": 3
		},
		"Username": "svc_access2@appsr-vr80.keyexample.com"
		}
		}
		
		It contains three IDs:
		• ID 2: The service account's ID.
		• ID 7: The service account user's ID.

Name	In	Description
		• ID 36: The ID of the service account user's key. Use this one to request the key.
IncludePrivateKey	Query	A Boolean that sets whether to include the private key of the SSH key pair in the response (True) or not (False). The default is *False*. If set to True, the X-Keyfactor-Key-Passphrase header must be supplied.
X-Keyfactor-Key-Passphrase	Header	**Required***. A string that sets a password used to secure the private key of the SSH key pair for download. This field is **required** if *IncludePrivateKey* is set to *True*.

Table 703: GET SSH Service Accounts Key {id} Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH service account. This ID is automatically set by Keyfactor Command.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair for the SSH service account.
PrivateKey	A string indicating the private key of the key pair for the SSH service account.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string indicating the date, in UTC, on which the SSH key pair was created.
StaleDate	A string indicating the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by the *Key Lifetime (days)* application setting. See Application Settings: SSH Tab on page 620 in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the administrator or group of administrators responsible for managing the key. This email address is used to alert the administrator or group of administrators when the key pair is approaching the end of its lifetime.
Comments	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/ServiceAccounts method will contain only one string in the array.
LogonCount	An integer indicating the number of Linux logons associated with the SSH key pair.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

DELETE SSH Service Accounts

The DELETE /SSH/ServiceAccounts method is used to delete one or more SSH service accounts in Keyfactor Command, including their SSH key pairs. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 704: DELETE SSH Service Accounts Input Parameters

Name	In	Description

ids	Body	**Required**. An array of integers indicating the Keyfactor Command reference IDs for the SSH service accounts to be deleted provided in the request body in the following format:
		`[4,12,17]`
		
		Use the *GET /SSH/ServiceAccounts* method (see GET SSH Service Accounts on the next page) to retrieve a list of all the SSH service accounts to determine the service accounts IDs.
		
		Tip: Be sure to use the ID of the service account itself and not the ID of the service account user or service account's key within the service account. For example, notice the following record returned from a GET /SSH/ServiceAccounts:
		<pre>{
		  "Id": 2,
		  "ClientHostname": "appsrvr80.keyexample.com",
		  "ServerGroup": {
		  "Id": "603d3d4c-89dd-4ab8-92e1-8e83db3d5546",
		  "GroupName": "Server Group Two",
		  "UnderManagement": false
		  },
		  "User": {
		  "Id": 7,
		  "Key": {
		  "Id": 36,
		  "Fingerprint":
		"kwuo2k3Ej7wFVMLhI3g+rxt2qXwGp7qcvzdBjVTDHNg=",
		  "PublicKey": "ssh-rsa
		AAAAB3NzaC1yc2EAAAADAQABAAABAQCAln+t [truncated for
		display]",
		  "KeyType": "RSA",
		  "KeyLength": 2048,
		  "CreationDate": "2020-11-17T17:53:55.68",
		  "Email": "pkiadmins@keyexample.com",
		  "Comments": [
		  "Access App Two"
		  ],
		  "LogonCount": 3
		  },
		  "Username": "svc_access2@appsrvr80.keyexample.com"
		}</pre>

Name	In	Description
		} It contains three IDs: • ID 2: The service account's ID. Use this one for delete requests. • ID 7: The service account user's ID. • ID 36: The ID of the service account user's key.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Service Accounts

The GET /SSH/ServiceAccounts method is used to retrieve one or more SSH service accounts defined in Keyfactor Command. Results can be limited to selected service accounts using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details for the requested SSH service accounts and their public keys. To return the SSH private key, use the GET /SSH/ServiceAccounts/Key/{id} method (see GET SSH Service Accounts Key ID on page 2234).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 705: GET SSH Service Accounts Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Service Account Key Search on page 553. The query fields supported for this endpoint are: • Comments (Key comments) • CreationDate • Id • KeyLength • KeyType • ServerGroup (Server Group ID) • ServerGroupName • Username
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Username*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 706: GET SSH Service Accounts Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH service account. This ID is automatically set by Keyfactor Command.
ClientHost-name	A string indicating the client hostname reference for the service account key. This field is used for reference only and does not need to match an actual client hostname. It is used when building the full user name of the service account key for mapping to Linux logons for publishing to Linux servers (e.g. username@client_hostname). The naming convention is to enter the hostname of the server on which the application that will use the private key resides (e.g. appsrvr12), but you can put anything you like in this field (e.g. cheesetoast).
Server-Group	An object that indicates the SSH server group for the service account. The server group is used to control who has access in Keyfactor Command to the service account key. It does not limit where the key can be published. See SSH Permissions on page 597 in the *Keyfactor Command Reference Guide* for more information. Server group information includes:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the SSH server group.
Owner	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information.

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group.

GroupName	A string indicating the name of the SSH server group.
SyncSchedul-e	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr><tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.<table><tr><th>Name</th><th>Description</th></tr><tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr></table>For example, every hour:<pre>"Interval": {
	  "Minutes": 60
	}</pre></td></tr><tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr></table>For example, daily at 11:30 pm:<pre>"Daily": {</pre></td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td></td><td><pre>
   "Time": "2023-11-25T23:30:00Z"
}
</pre></td></tr>
<tr><td>Week-ly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:m-m:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>Under-Management</td><td>A Boolean indicating whether the SSH server group is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
</table>
</td></tr>
<tr><td>User</td><td>An object containing information about the service account user. Service account user details include:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SSH service account user.</td></tr><tr><td>Key</td><td>An object containing information about the key for the service account user. Key details include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SSH service account's key.</td></tr><tr><td>Fingerprint</td><td>A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.</td></tr><tr><td>PublicKey</td><td>A string indicating the public key of the key pair for the SSH service account.</td></tr><tr><td>KeyType</td><td>A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: - RSA - ECDSA - Ed25519 </td></tr><tr><td>KeyLength</td><td>An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.</td></tr><tr><td>CreationDate</td><td>A string indicating the date, in UTC, on which the SSH key pair was created.</td></tr><tr><td>StaleDate</td><td>A string indicating the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by</td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the Key Lifetime (days) application setting. See <a>Application Settings: SSH Tab on page 620 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Email</td><td>A string containing the email address of the administrator or group of administrators responsible for managing the key. This email address is used to alert the administrator or group of administrators when the key pair is approaching the end of its lifetime.</td></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the service account. The username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Service Accounts

The POST /SSH/ServiceAccounts method is used to create a new SSH service account in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the new SSH service account.

Before adding a new SSH service account, be sure that you have added at least one server group (see POST SSH Server Groups on page 2202) and that your Keyfactor Bash Orchestrator has been registered and approved in Keyfactor Command (see GET Agents on page 858).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 707: POST SSH Service Accounts Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>KeyGenerationRequest</td><td>Body</td><td>Required. An object that set the information to include in the SSH key pair request. Key generation request details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>KeyType</td><td>Required. A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are:
<table>
<tr><th>Numeric Value</th><th>Text Value</th></tr>
<tr><td>1</td><td>ECDSA</td></tr>
<tr><td>2</td><td>Ed25519</td></tr>
<tr><td>3</td><td>RSA</td></tr>
</table>
The KeyType may be specified using either the numeric value or text value.</td></tr>
<tr><td>PrivateKeyFormat</td><td>Required. A string indicating the format to use for the downloadable private key. Possible values are:
<table>
<tr><th>Numeric Value</th><th>Text Value</th></tr>
<tr><td>1</td><td>OpenSSH</td></tr>
<tr><td>2</td><td>PKCS8</td></tr>
</table>
The PrivateKeyFormat may be specified using either the numeric value or text value.</td></tr>
<tr><td>KeyLength</td><td>Required*. An integer indicating the key length for the SSH key. The key length supported depends on the key</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA. This field is optional if the KeyType is set to ECDSA or Ed25519 and required if the KeyType is set to RSA.</td></tr>
<tr><td>Email</td><td>Required. A string containing the email address of the administrator or group of administrators responsible for managing the key. This email address is used to alert the administrator or group of administrators when the key pair is approaching the end of its lifetime.</td></tr>
<tr><td>Password</td><td>Required. A string that sets a password used to secure the private key of the SSH key pair for download.</td></tr>
<tr><td>Comment</td><td>A string containing the user-defined descriptive comment, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks.</td></tr>
</table>
</td></tr>
<tr><td>User</td><td>Body</td><td>Required. An object containing information about the service account user. User details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Username</td><td>Required. A string indicating the short name of the SSH service account user (e.g. myapp). This, together with the ClientHostname, is used to build the full user name (e.g.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>myapp@appsrvr75).</td></tr>
<tr><td>LogonIds</td><td>An array of integers indicating the Keyfactor Command reference IDs of Linux logons that should be associated with the service account in order to publish the service account's public key to the servers on which the logons are located.</td></tr>
</table>
</td></tr>
<tr><td>ClientHostname</td><td>Body</td><td>Required. A string indicating the client hostname reference for the service account key. This field is used for reference only and does not need to match an actual client hostname. It is used when building the full user name of the service account key for mapping to Linux logons for publishing to Linux servers (e.g. user-name@client_hostname). The naming convention is to use the host-name of the server on which the application that will use the private key resides (e.g. appsrvr12), but you can put anything you like in this field (e.g. cheesetoast).</td></tr>
<tr><td>ServerGroupId</td><td>Body</td><td>Required. A string indicating the Keyfactor Command reference GUID for the SSH server group for the service account. The server group is used to control who has access in Keyfactor Command to the service account key. It does not limit where the key can be published. See SSH Permissions on page 597 in the Keyfactor Command Reference Guide for more information.</td></tr>
</table>

Table 708: POST SSH Service Accounts Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH service account. This ID is automatically set by Keyfactor Command.
ClientHost-name	A string indicating the client hostname reference for the service account key. This field is used for reference only and does not need to match an actual client hostname. It is used when building the full user name of the service account key for mapping to Linux logons for publishing to Linux servers (e.g. username@client_hostname). The naming convention is to enter the hostname of the server on which the application that will use the private key resides (e.g. appsrvr12), but you can put anything you like in this field (e.g. cheesetoast).
Server-Group	An object that indicates the SSH server group for the service account. The server group is used to control who has access in Keyfactor Command to the service account key. It does not limit where the key can be published. See SSH Permissions on page 597 in the *Keyfactor Command Reference Guide* for more information. Server group information includes:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the SSH server group.
Owner	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information.

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group.

GroupName	A string indicating the name of the SSH server group.
SyncSchedul-e	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr><tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.<table><tr><th>Name</th><th>Description</th></tr><tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr></table>For example, every hour:<pre>"Interval": {
	  "Minutes": 60
	}</pre></td></tr><tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr></table>For example, daily at 11:30 pm:<pre>"Daily": {</pre></td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Nam-e</th><th>Description</th></tr>
<tr><td></td><td><pre>
   "Time": "2023-11-25T23:30:00Z"
}
</pre></td></tr>
<tr><td>Week-ly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:m-m:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>Under-Management</td><td>A Boolean indicating whether the SSH server group is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
</table>
</td></tr>
<tr><td>User</td><td>An object containing information about the service account user. Service account user details include:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SSH service account user.</td></tr><tr><td>Key</td><td>An object containing information about the key for the service account user. Key details include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SSH service account's key.</td></tr><tr><td>Fingerprint</td><td>A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.</td></tr><tr><td>PublicKey</td><td>A string indicating the public key of the key pair for the SSH service account.</td></tr><tr><td>KeyType</td><td>A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: - RSA - ECDSA - Ed25519 </td></tr><tr><td>KeyLength</td><td>An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.</td></tr><tr><td>CreationDate</td><td>A string indicating the date, in UTC, on which the SSH key pair was created.</td></tr><tr><td>StaleDate</td><td>A string indicating the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by</td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the Key Lifetime (days) application setting. See <a>Application Settings: SSH Tab on page 620 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Email</td><td>A string containing the email address of the administrator or group of administrators responsible for managing the key. This email address is used to alert the administrator or group of administrators when the key pair is approaching the end of its lifetime.</td></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the service account. The username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
<tr><td>LogonIds</td><td>An array of integers indicating the Keyfactor Command reference IDs of Linux logons that are associated with the service account in order to publish the service account's public key to the servers on which the logons are located.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT SSH Service Accounts

The PUT /SSH/ServiceAccounts method is used to update an existing SSH service account in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the SSH service account.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 709: PUT SSH Service Accounts Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>KeyUpdateRequest</td><td>Body</td><td>Required. An object that sets the information to include in the SSH service account key update request. Key update request information includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>Required. An integer indicating the Keyfactor Command reference ID for the service account's key.</td></tr><tr><td>Email</td><td>Required. A string containing the email address of the administrator or group of administrators responsible for managing the key. This email address is used to alert the administrator or group of administrators when the key pair is approaching the end of its lifetime.</td></tr><tr><td>Comment</td><td>A string containing the user-defined descriptive comment, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks.</td></tr></table></td></tr>
<tr><td>Id</td><td>Body</td><td>Required. An integer indicating the Keyfactor Command reference ID for the service account.
Use the GET /SSH/ServiceAccounts method (see GET SSH Service Accounts on page 2240) to retrieve a list of all the SSH service accounts to determine the service account's ID.</td></tr>
</table>

Table 710: PUT SSH Service Accounts Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH service account. This ID is automatically set by Keyfactor Command.
ClientHost-name	A string indicating the client hostname reference for the service account key. This field is used for reference only and does not need to match an actual client hostname. It is used when building the full user name of the service account key for mapping to Linux logons for publishing to Linux servers (e.g. username@client_hostname). The naming convention is to enter the hostname of the server on which the application that will use the private key resides (e.g. appsrvr12), but you can put anything you like in this field (e.g. cheesetoast).
Server-Group	An object that indicates the SSH server group for the service account. The server group is used to control who has access in Keyfactor Command to the service account key. It does not limit where the key can be published. See SSH Permissions on page 597 in the *Keyfactor Command Reference Guide* for more information. Server group information includes:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the SSH server group.
Owner	An object indicating the Active Directory user who owns the server group. See SSH Server Groups on page 560 in the *Keyfactor Command Reference Guide* for more information.

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the *user* who holds the owner role on the SSH server group.

GroupName	A string indicating the name of the SSH server group.
SyncSchedul-e	An object providing the inventory schedule for the SSH server group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr><tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.<table><tr><th>Name</th><th>Description</th></tr><tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr></table>For example, every hour:<pre>"Interval": {
	  "Minutes": 60
	}</pre></td></tr><tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr></table>For example, daily at 11:30 pm:<pre>"Daily": {</pre></td></tr></table></td></tr></table>

Name	Description

Name	Description

Nam-e	Description

	<pre>"Time": "2023-11-25T23:30:00Z"
	}</pre>

Week-ly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).
Days	An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).

For example, every Monday, Wednesday and Friday at 5:30 pm:

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the Keyfactor API Reference and Utility—Swagger—Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
</td></tr>
<tr><td>Under-Management</td><td>A Boolean indicating whether the SSH server group is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
</table>
</td></tr>
<tr><td>User</td><td>An object containing information about the service account user. Service account user details include:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SSH service account user.</td></tr><tr><td>Key</td><td>An object containing information about the key for the service account user. Key details include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SSH service account's key.</td></tr><tr><td>Fingerprint</td><td>A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.</td></tr><tr><td>PublicKey</td><td>A string indicating the public key of the key pair for the SSH service account.</td></tr><tr><td>KeyType</td><td>A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: - RSA - ECDSA - Ed25519 </td></tr><tr><td>KeyLength</td><td>An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.</td></tr><tr><td>CreationDate</td><td>A string indicating the date, in UTC, on which the SSH key pair was created.</td></tr><tr><td>StaleDate</td><td>A string indicating the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by</td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the Key Lifetime (days) application setting. See <a>Application Settings: SSH Tab on page 620 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Email</td><td>A string containing the email address of the administrator or group of administrators responsible for managing the key. This email address is used to alert the administrator or group of administrators when the key pair is approaching the end of its lifetime.</td></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the service account. The username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
<tr><td>LogonIds</td><td>An array of integers indicating the Keyfactor Command reference IDs of Linux logons that are associated with the service account in order to publish the service account's public key to the servers on which the logons are located.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Service Accounts Rotate ID

The POST /SSH/ServiceAccounts/Rotate/{id} method is used to generate a new key pair in Keyfactor Command for an existing SSH service account. This method returns HTTP 200 OK on a success with details for the new key pair of the SSH service account, including the private key.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 711: GET SSH Service Accounts Rotate {id} Input Parameters

Name	In	Description

id	Path	**Required**. The Keyfactor Command reference ID for the SSH service account key for which to retrieve key information.
		Use the *GET /SSH/ServiceAccounts* method (see GET SSH Service Accounts on page 2240) to retrieve a list of all the SSH service accounts to determine the service account's key ID.
		
		Tip: Be sure to use the ID of the service account itself and not the ID of the service account user or service account's key within the service account. For example, notice the following record returned from a GET /SSH/ServiceAccounts:
		
		{
		"Id": 2,
		"ClientHostname": "appsrvr80.keyexample.com",
		"ServerGroup": {
		"Id": "603d3d4c-89dd-4ab8-92e1-8e83db3d5546",
		"GroupName": "Server Group Two",
		"UnderManagement": false
		},
		"User": {
		"Id": 7,
		"Key": {
		"Id": 36,
		"Fingerprint": "kwuo2k3Ej7wFVMLhI3g+rx-t2qXwGp7qcvzdBjVTDHNg=",
		"PublicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCAln+t [truncated for display]",
		"KeyType": "RSA",
		"KeyLength": 2048,
		"CreationDate": "2020-11-17T17:53:55.68",
		"Email": "pkiadmins@keyexample.com",
		"Comments": [
		"Access App Two"
		],
		"LogonCount": 3
		},
		"Username": "svc_access2@appsr-vr80.keyexample.com"
		}
		}
		
		It contains three IDs:
		• ID 2: The service account's ID. Use this one to rotate the key.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>

- ID 7: The service account user's ID.
- ID 36: The ID of the service account user's key.

</td></tr>
<tr><td>KeyType</td><td>Body</td><td>Required. A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>ECDSA</td></tr>
<tr><td>2</td><td>Ed25519</td></tr>
<tr><td>3</td><td>RSA</td></tr>
</table>
</td></tr>
<tr><td>PrivateKeyFormat</td><td>Body</td><td>Required. A string indicating the format to use for the downloadable private key. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>OpenSSH</td></tr>
<tr><td>2</td><td>PKCS8</td></tr>
</table>
</td></tr>
<tr><td>KeyLength</td><td>Body</td><td>Required*. An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA. This field is optional if the KeyType is set to ECDSA or Ed25519 and required if the KeyType is set to RSA.</td></tr>
<tr><td>Email</td><td>Body</td><td>Required. A string containing the email address of the administrator or group of administrators responsible for managing the key. This email address is used to alert the administrator or group of administrators when the key pair is approaching the end of its lifetime.</td></tr>
<tr><td>Password</td><td>Body</td><td>Required. A string that sets a password used to secure the private key of the SSH key pair for download.</td></tr>
<tr><td>Comment</td><td>Body</td><td>An string containing the user-defined descriptive comment, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks.</td></tr>
</table>

Table 712: GET SSH Service Accounts Rotate {id} Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID for the SSH service account key. This ID is automatically set by Keyfactor Command.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair for the SSH service account.
PrivateKey	A string indicating the private key of the key pair for the SSH service account.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
Comments	An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/ServiceAccounts method will contain only one string in the array.
LogonCount	An integer indicating the number of Linux logons associated with the SSH key pair.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.34.6 SSH Users

The SSH Users component of the Keyfactor API includes methods necessary to retrieve, create, update, rotate, and delete users and associated keys in Keyfactor Command.

Table 713: SSH Users Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the SSH user with the specified ID.	DELETE SSH Users ID below
/{id}	GET	Returns the SSH user with the specified ID.	GET SSH Users ID on the next page
/	GET	Returns a list of SSH users based on the specified filters.	GET SSH Users on page 2276
/	POST	Creates a new SSH user.	POST SSH Users on page 2287
/	PUT	Updates an existing SSH user.	PUT SSH Users on page 2289
/Access	POST	Creates a mapping from the SSH user to one or more Linux logons.	POST SSH Users Access on page 2290

DELETE SSH Users ID

The DELETE /SSH/Users/{id} method is used to delete an SSH user in Keyfactor Command. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/

Table 714: DELETE SSH Users {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH user (user or service account) to be deleted. Use the *GET /SSH/Users* method (see GET SSH Users on page 2276) to retrieve a list of all the SSH users to determine the user's ID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Users ID

The GET /SSH/Users/{id} method is used to retrieve an SSH user defined in Keyfactor Command.The method can return either a *user* or a *service account* See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information on the difference between *users* and *service accounts*. This method returns HTTP 200 OK on a success with details for the requested SSH user and its public key. To return an SSH private key, use the GET /SSH/Keys/MyKey method (see GET SSH Keys My Key on page 2130) for a user account or the GET /SSH/ServiceAccounts/Key/{id} method (see GET SSH Service Accounts Key ID on page 2234) for a service account.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which user to logon mappings are associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the GET /SSH/Users/{id} method redesigns how logon information for the user is returned, providing a greater level of detail in the returned data.

Table 715: GET SSH Users {id} v2 Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH user (user or service account) to be retrieved. Use the *GET /SSH/Users* method (see GET SSH Users on page 2276) to retrieve a list of all the SSH users to determine the user's ID.

Table 716: GET SSH Users {id} v2 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user.
Key	An object containing information about the key for the user. Key details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user's key.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair for the SSH user.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string containing the date, in UTC, on which the SSH key pair was created.
StaleDate	A string containing the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by the *Key Lifetime (days)* application setting. See Application Settings: SSH Tab on page 620 in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the user, for user accounts, or administrator or group of administrators responsible for managing the key, for service accounts. This email address is used to alert the user or administrator when the key pair is approaching the end of its lifetime.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/Keys/MyKey or POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the user or service account. For a user account, the username appears in DOMAIN\\username format (e.g. KEYEXAMPLE\\jsmith). For a service account, the username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
<tr><td>Access</td><td>An object containing information about the Linux logons mapped to the user. Linux logon mapping details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the Linux logon.</td></tr>
<tr><td>KeyCount</td><td>An integer indicating the number of SSH keys associated with the Linux logon.</td></tr>
<tr><td>Access</td><td>An object containing information about the users mapped to the Linux logon.</td></tr>
</table>
</td></tr>
<tr><td>IsGroup</td><td>A Boolean indicating whether the user is an Active Directory group (true) or not (false).</td></tr>
</table>

Version 1

Version 1 of the GET /SSH/Users/{id} method includes the same capabilities as version 2, but offers more limited information on returned logons for the user.

Table 717: GET SSH Users {id} v1 Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the Keyfactor Command reference ID for the SSH user (user or service account) to be retrieved. Use the *GET /SSH/Users* method (see GET SSH Users on page 2276) to retrieve a list of all the SSH users to determine the user's ID.

Table 718: GET SSH Users {id} v1 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user.
Key	An object containing information about the key for the user. Key details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user's key.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair for the SSH user.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string containing the date, in UTC, on which the SSH key pair was created.
StaleDate	A string containing the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by the *Key Lifetime (days)* application setting. See Application Settings: SSH Tab on page 620 in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the user, for user accounts, or administrator or group of administrators responsible for managing the key, for service accounts. This email address is used to alert the user or administrator when the key pair is approaching the end of its lifetime.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/Keys/MyKey or POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the user or service account. For a user account, the username appears in DOMAIN\\username format (e.g. KEYEXAMPLE\\jsmith). For a service account, the username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
<tr><td>LogonIds</td><td>An array of integers indicating the Keyfactor Command reference IDs for the Linux logons mapped to the user to cause the user's SSH public key to be published out to the Linux servers on which those logons reside.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

GET SSH Users

The GET /SSH/Users method is used to retrieve one or more SSH users defined in Keyfactor Command. The method returns both *users* and *service accounts*. See SSH on page 525 in the *Keyfactor Command Reference Guide* for more information on the difference between *users* and *service accounts*. Results can be limited to selected users using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details for the requested SSH users and their public keys. To return the SSH private key, use the GET /SSH/Keys/MyKey method (see GET SSH Keys My Key on page 2130) for user accounts and the GET /SSH/ServiceAccounts/Key/{id} method (see GET SSH Service Accounts Key ID on page 2234) for service accounts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which user to logon mappings are associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 852.

Version 2

Version 2 of the GET /SSH/Users method redesigns how logon information for the user is returned, providing a greater level of detail in the returned data.

Table 719: GET SSH Users v2 Input Parameters

Name	In	Description
showOwnedAccess	Query	A Boolean that specifies whether to return only users that have logons on servers that the requesting user owns (*true*) or not (*false*). The default is *false*. This option applies only to requesting users with *SSH User* or *SSH Server Admin* permissions; users with *SSH Enterprise Admin* permissions will see all users regardless of the configuration of this setting. Use the *GET /SSH/Servers* method (see GET SSH Servers on page 2162) or the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 2197) to determine ownership of a server or server group. **Example:** Example Scenario One • Server A is owned by Gina and server B is owned by John. • Gina is an *SSH Server Admin* but not an *SSH Enterprise Admin*. • Dave has a logon on server B but not on server A. Gina does a GET /SSH/Users with *showOwnedAccess=false* and looks at the results for Dave's user record. She sees Dave's user record, but sees no specific logon information for Dave (other than the LogonCount), because all Dave's logons are on servers that Gina does not own. Gina does a GET /SSH/Users with *showOwnedAccess=true* and looks at the results for Dave's user record. Dave's user record does not appear. The presence or absence of Dave's user record is controlled by *showOwnedAccess*. The presence or absence of logon information associated with Dave's user record is controlled by Gina's level of SSH permissions—with *SSH Server Admin* permissions, Gina will always see only logons for servers that she owns. **Example:** Example Scenario Two • Server A is owned by Gina and server B is owned by John. • Gina is an *SSH Server Admin* but not an *SSH Enterprise Admin*. • Dave has a logon on server B and a logon on server A.

Name	In	Description
		Gina does a GET /SSH/Users with *showOwnedAccess=false* and looks at the results for Dave's user record. She sees Dave's user record and she sees logon information for server A, but no logon information for server B. Because Gina does not own server B, logon information for that server is not visible to her. Gina does a GET /SSH/Users with *showOwnedAccess=true* and looks at the results for Dave's user record. She sees Dave's user record and she sees logon information for server A, but no logon information for server B. Because Gina does not own server B, logon information for that server is not visible to her. Notice there is no difference here in the results whether you choose *true* or *false* because at least one logon for Dave is present on a server owned by Gina. The *showOwnedAccess* option only comes into play when a user has no logons on a server owned by the requesting user. The presence or absence of logon information associated with Dave's user record is controlled by Gina's level of SSH permissions—with *SSH Server Admin* permissions, Gina will always see only logons for servers that she owns.
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, refer to: Using the SSH Server Search on page 583. The query fields supported for this endpoint are: • Email • Fingerprint • IsServiceAccount • KeyLength • KeyType • LogonCount • LogonServerGroupId • LogonServerId • ServiceAccountId • StaleDate

Name	In	Description
		• Username
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Username*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 720: GET SSH Users v2 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user.
Key	An object containing information about the key for the user. Key details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user's key.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair for the SSH user.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string containing the date, in UTC, on which the SSH key pair was created.
StaleDate	A string containing the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by the *Key Lifetime (days)* application setting. See Application Settings: SSH Tab on page 620 in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the user, for user accounts, or administrator or group of administrators responsible for managing the key, for service accounts. This email address is used to alert the user or administrator when the key pair is approaching the end of its lifetime.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/Keys/MyKey or POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the user or service account. For a user account, the username appears in DOMAIN\\username format (e.g. KEYEXAMPLE\\jsmith). For a service account, the username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
<tr><td>Access</td><td>An object containing information about the Linux logons mapped to the user. Linux logon mapping details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the Linux logon.</td></tr>
<tr><td>KeyCount</td><td>An integer indicating the number of SSH keys associated with the Linux logon.</td></tr>
<tr><td>Access</td><td>An object containing information about the users mapped to the Linux logon.</td></tr>
</table>
</td></tr>
<tr><td>IsGroup</td><td>A Boolean indicating whether the user is an Active Directory group (true) or not (false).</td></tr>
</table>

Version 1

Version 1 of the GET /SSH/Users method includes the same capabilities as version 2, but offers more limited information on returned logons for the user.

Table 721: GET SSH Users v1 Input Parameters

Name	In	Description
showOwnedAccess	Query	A Boolean that specifies whether to return only users that have logons on servers that the requesting user owns (*true*) or not (*false*). The default is *false*. This option applies only to requesting users with *SSH User* or *SSH Server Admin* permissions; users with *SSH Enterprise Admin* permissions will see all users regardless of the configuration of this setting. Use the *GET /SSH/Servers* method (see GET SSH Servers on page 2162) or the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 2197) to determine ownership of a server or server group. **Example:** Example Scenario One • Server A is owned by Gina and server B is owned by John. • Gina is an *SSH Server Admin* but not an *SSH Enterprise Admin*. • Dave has a logon on server B but not on server A. Gina does a GET /SSH/Users with *showOwnedAccess=false* and looks at the results for Dave's user record. She sees Dave's user record, but sees no specific logon information for Dave (other than the LogonCount), because all Dave's logons are on servers that Gina does not own. Gina does a GET /SSH/Users with *showOwnedAccess=true* and looks at the results for Dave's user record. Dave's user record does not appear. The presence or absence of Dave's user record is controlled by *showOwnedAccess*. The presence or absence of logon information associated with Dave's user record is controlled by Gina's level of SSH permissions—with *SSH Server Admin* permissions, Gina will always see only logons for servers that she owns. **Example:** Example Scenario Two • Server A is owned by Gina and server B is owned by John. • Gina is an *SSH Server Admin* but not an *SSH Enterprise Admin*. • Dave has a logon on server B and a logon on server A.

Name	In	Description
		Gina does a GET /SSH/Users with *showOwnedAccess=false* and looks at the results for Dave's user record. She sees Dave's user record and she sees logon information for server A, but no logon information for server B. Because Gina does not own server B, logon information for that server is not visible to her. Gina does a GET /SSH/Users with *showOwnedAccess=true* and looks at the results for Dave's user record. She sees Dave's user record and she sees logon information for server A, but no logon information for server B. Because Gina does not own server B, logon information for that server is not visible to her. Notice there is no difference here in the results whether you choose *true* or *false* because at least one logon for Dave is present on a server owned by Gina. The *showOwnedAccess* option only comes into play when a user has no logons on a server owned by the requesting user. The presence or absence of logon information associated with Dave's user record is controlled by Gina's level of SSH permissions—with *SSH Server Admin* permissions, Gina will always see only logons for servers that she owns.
queryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns for the same feature. For querying guidelines, refer to: Using the SSH Server Search on page 583. The query fields supported for this endpoint are: • Email • Fingerprint • IsServiceAccount • KeyLength • KeyType • LogonCount • LogonServerGroupId • LogonServerId • ServiceAccountId • StaleDate

Name	In	Description
		• Username
pageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
returnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
sortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Username*.
sortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 722: GET SSH Users v1 Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user.
Key	An object containing information about the key for the user. Key details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user's key.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair for the SSH user.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string containing the date, in UTC, on which the SSH key pair was created.
StaleDate	A string containing the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by the *Key Lifetime (days)* application setting. See Application Settings: SSH Tab on page 620 in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the user, for user accounts, or administrator or group of administrators responsible for managing the key, for service accounts. This email address is used to alert the user or administrator when the key pair is approaching the end of its lifetime.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/Keys/MyKey or POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the user or service account. For a user account, the username appears in DOMAIN\\username format (e.g. KEYEXAMPLE\\jsmith). For a service account, the username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
<tr><td>LogonIds</td><td>An array of integers indicating the Keyfactor Command reference IDs for the Linux logons mapped to the user to cause the user's SSH public key to be published out to the Linux servers on which those logons reside.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Users

The POST /SSH/Users method is used to create a new SSH user in Keyfactor Command and, optionally, associate the user with one or more Linux logons during creation to allow the public key for the user to be published out to a Linux server—for servers in *inventory and publish policy* mode. This method returns HTTP 200 OK on a success with the details of the user to logon mapping, if any.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/

SSH actions are affected by ownership on the server group with which user to logon mappings are associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 723: POST SSH Users Input Parameters

Name	In	Description
Username	Body	**Required**. A string indicating the full username of the *user* or *service account*. For a *user* account, the username is given in DOMAIN\\username format (e.g. KEYEXAMPLE\\jsmith). For a *service account*, the username is made up of a user name (e.g. svc_myapp) and client hostname reference for the service account. The client hostname is used for reference only and does not need to match an actual client hostname. The naming convention is to enter the hostname of the server on which the application that will use the private key resides (e.g. appsrvr12), but you can put anything you like in this field (e.g. cheesetoast). The full service account name is given in the form username@clienthostname (e.g. svc_myapp@appsrvr75).
LogonIds	Body	An array of integers indicating the Keyfactor Command reference IDs for the Linux logons to map to the user to cause the user's SSH public key to be published out to the Linux servers on which those logons reside. These are provided in the following format: `[12,27,39]` Use the *GET /SSH/Logons* method (see GET SSH Logons on page 2147) to retrieve a list of all the SSH logons to determine the logon's ID(s).

Table 724: POST SSH Users Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID of the SSH user.
Username	A string indicating the full username of the *user* or *service account*.
LogonIds	An array of integers indicating the Keyfactor Command reference IDs for the Linux logons to map to the user to cause the user's SSH public key to be published out to the Linux servers on which those logons reside.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

PUT SSH Users

The PUT /SSH/Users method is used to update an existing SSH user in Keyfactor Command and, optionally, associate the user with one or more Linux logons to allow the public key for the user to be published out to a Linux server—for servers in *inventory and publish policy* mode. This method returns HTTP 200 OK on a success with the details of the user to logon mapping, if any.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/

SSH actions are affected by ownership on the server group with which user to logon mappings are associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Table 725: PUT SSH Users Input Parameters

Name	In	Description
ID	Body	**Required**. An integer indicating the Keyfactor Command reference ID of the SSH user. Use the *GET /SSH/Users* method (see GET SSH Users on page 2276) to retrieve a list of all the SSH users to determine the user's ID.
LogonIds	Body	An array of integers indicating the Keyfactor Command reference IDs for the Linux logons to map to the user to cause the user's SSH public key to be published out to the Linux servers on which those logons reside. These are provided in the following format: `[12,27,39]` Use the *GET /SSH/Logons* method (see GET SSH Logons on page 2147) to retrieve a list of all the SSH logons to determine the logon's ID(s). **Important:** Logon IDs you provide here replace any existing logon IDs associated with the user. To avoid accidentally removing access for users, check existing logons for the user (see GET SSH Users on page 2276) before updating and provide both existing and new logon IDs.

Table 726: POST SSH Users Response Data

Name	Description
ID	An integer indicating the Keyfactor Command reference ID of the SSH user.
Username	A string indicating the full username of the *user* or *service account*.
LogonIds	An array of integers indicating the Keyfactor Command reference IDs for the Linux logons to map to the user to cause the user's SSH public key to be published out to the Linux servers on which those logons reside.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

POST SSH Users Access

The POST /SSH/Users/Access method is used to create a mapping of one or more Linux logons to a Keyfactor Command user or service account. This method returns HTTP 200 OK on a success with the details of the user to logon mapping, if any.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssh/server_admin/
OR
/ssh/enterprise_admin/
SSH actions are affected by ownership on the server group with which user to logon mappings are associated and limited for users with only the *Server Admin* (/ssh/server_admin/) role. For more information, see SSH Permissions on page 597.

Tip: Before creating a logon to user mapping, be sure that you have switched the server to which you will add your mapping (or its server group) to *inventory and publish policy* mode so that the key for the user will be published to the server. If the server is in *inventory only* mode and you add a mapping for it in Keyfactor Command, the mapping will appear in Keyfactor Command only and the key for the user will not be published out to the server.

Table 727: POST SSH Users Access Input Parameters

Name	In	Description
ID	Body	**Required**. An integer indicating the Keyfactor Command reference ID of the SSH user. Use the *GET /SSH/Users* method (see GET SSH Users on page 2276) to retrieve a list of all the SSH users to determine the user's ID.
LogonIds	Body	An array of integers indicating the Keyfactor Command reference IDs for the Linux logons to map to the user to cause the user's SSH public key to be published out to the Linux servers on which those logons reside. These are provided in the following format: `[12,27,39]` Use the *GET /SSH/Logons* method (see GET SSH Logons on page 2147) to retrieve a list of all the SSH logons to determine the logon's ID(s). **Important:** Logon IDs you provide here replace any existing logon IDs associated with the user. To avoid accidentally removing access for users, check existing logons for the user (see GET SSH Users on page 2276) before updating and provide both existing and new logon IDs.

Table 728: POST SSH Users Access Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user.
Key	An object containing information about the key for the user. Key details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH user's key.
Fingerprint	A string indicating the fingerprint of the public key. Each SSH public key has a single cryptographic fingerprint that can be used to uniquely identify the key.
PublicKey	A string indicating the public key of the key pair for the SSH user.
KeyType	A string indicating the cryptographic algorithm used to generate the SSH key. Possible values are: • RSA • ECDSA • Ed25519
KeyLength	An integer indicating the key length for the SSH key. The key length supported depends on the key type selected. Keyfactor Command supports 256 bits for Ed25519 and ECDSA and 2048 or 4096 bits for RSA.
CreationDate	A string containing the date, in UTC, on which the SSH key pair was created.
StaleDate	A string containing the date, in UTC, after which the SSH key pair is considered to be out of date based on the key lifetime defined by the *Key Lifetime (days)* application setting. See Application Settings: SSH Tab on page 620 in the *Keyfactor Command Reference Guide* for more information.
Email	A string containing the email address of the user, for user accounts, or administrator or group of administrators responsible for managing the key, for service accounts. This email address is used to alert the user or administrator when the key pair is approaching the end of its lifetime.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Comments</td><td>An array of strings containing one or more strings with the user-defined descriptive comments, if any, on the key. Although entry of an email address in the comment field of an SSH key is traditional, this is not a required format. The comment may can contain any characters supported for string fields, including spaces and most punctuation marks. Keys created through the Keyfactor Command Management Portal or with the POST /SSH/Keys/MyKey or POST /SSH/ServiceAccounts method will contain only one string in the array.</td></tr>
<tr><td>LogonCount</td><td>An integer indicating the number of Linux logons associated with the SSH key pair.</td></tr>
</table>
</td></tr>
<tr><td>Username</td><td>A string indicating the full username of the user or service account. For a user account, the username appears in DOMAIN\\username format (e.g. KEYEXAMPLE\\jsmith). For a service account, the username is made up of the user name and ClientHostname entered when the service account is created (e.g. myapp@appsrvr75).</td></tr>
<tr><td>Access</td><td>An object containing information about the Linux logons mapped to the user. Linux logon mapping details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the Linux logon.</td></tr>
<tr><td>Username</td><td>A string indicating the user's logon name on the Linux server.</td></tr>
<tr><td>KeyCount</td><td>An integer indicating the number of SSH keys associated with the Linux logon.</td></tr>
<tr><td>Access</td><td>An object containing information about the users mapped to the Linux logon.</td></tr>
</table>
</td></tr>
<tr><td>IsGroup</td><td>A Boolean indicating whether the user is an Active Directory group (true) or not (false).</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.35 SMTP

The SMTP component of the Keyfactor API includes methods necessary to programmatically edit and retrieve the SMTP configuration profile and send a test email message. Editing the SMTP configuration profile in Keyfactor Command will only apply within the software. Only one SMTP profile may be configured.

Table 729: SMTP Endpoints

Endpoint	Method	Description	Link
/	GET	Returns information about the SMTP configuration profile.	GET SMTP below
/	PUT	Updates settings for the SMTP configuration profile.	PUT SMTP on page 2296
/Test	POST	Sends a test email message to confirm SMTP configuration.	POST SMTP Test on page 2299

3.6.35.1 GET SMTP

The GET /SMTP method is used to retrieve the SMTP configuration profile from Keyfactor Command. This method returns HTTP 200 OK on a success with details about the SMTP profile. Only one profile may be configured. There are no input parameters for this method.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/system_settings/read/

Table 730: GET SMTP Response Data

Name	Description
Host	A string indicating the fully qualified domain name of your SMTP host (e.g. corpexch02.keyexample.com).
Id	An integer indicating the Keyfactor Command reference ID of the SMTP record. This will be 1 in most environments.
Port	An integer indicating the SMTP port (e.g. 25).
RelayAuthenticationType	An integer indicating the type of authentication used to connect to the mail server. Possible values are: **Value** / **Description** 0 – Anonymous 2 – Explicit Credentials
RelayUsername	A string indicating the username of the account providing authentication to the mail server if *RelayAuthenticationType* is set to 2. The username should be provided in DOMAIN\\username format. For most mail server configurations, the username provided must have as a valid email address the email address you set in the *SenderAccount* parameter.
SenderAccount	A string indicating the sender for email messages delivered from Keyfactor Command, in the form of an email address (e.g. jsmith@keyexample.com). Depending on the email configuration in your environment, the sender account may need to be a valid user on your mail server or you may be able to put anything in this field.
SenderAddress	A string indicating the sender for email messages delivered from Keyfactor Command, in the form of an email address (e.g. jsmith@keyexample.com). This is considered deprecated and may be removed in a future release.
SenderName	A string indicating the name that appears as the "from" in the user's mail client (e.g. Keyfactor Command). This value is used for both configurations of *RelayAuthenticationType*.
UseSSL	A Boolean indicating that mail should be delivered over TLS/SSL. Not all mail servers support this.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.35.2 PUT SMTP

The PUT /SMTP method is used to update the SMTP configuration profile information. This method returns HTTP 200 OK on a success with details about the SMTP configuration profile.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/system_setting/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 731: PUT SMTP Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Host</td><td>Body</td><td>Required. A string indicating the fully qualified domain name of your SMTP host (e.g. corpexch02.keyexample.com).</td></tr>
<tr><td>Id</td><td>Body</td><td>Required. An integer indicating the Keyfactor Command reference ID of the SMTP record. This will be 1 in most environments.</td></tr>
<tr><td>Port</td><td>Body</td><td>An integer indicating the SMTP port (e.g. 25).</td></tr>
<tr><td>RelayAuthenticationType</td><td>Body</td><td>An integer indicating the type of authentication used to connect to the mail server. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Anonymous</td></tr><tr><td>2</td><td>Explicit Credentials</td></tr></table></td></tr>
<tr><td>RelayPassword</td><td>Body</td><td>Required*. A string indicating the password of the user specified by RelayUsername if RelayAuthenticationType is set to 2. This field is required if RelayAuthenticationType is set to 2.
No data is output in this field on a GET.</td></tr>
<tr><td>RelayUsername</td><td>Body</td><td>Required*. A string indicating the username of the account providing authentication to the mail server if RelayAuthenticationType is set to 2. The username should be provided in DOMAIN\\username format. This field is required if RelayAuthenticationType is set to 2.
For most mail server configurations, the username provided must have as a valid email address the email address you set in the SenderAccount parameter.</td></tr>
<tr><td>SenderAccount</td><td>Body</td><td>Required. A string indicating the sender for email messages delivered from Keyfactor Command, in the form of an email address (e.g. jsmith@keyexample.com). Depending on the email configuration in your environment, the sender account may need to be a valid user on your mail server or you may be able to put anything in this field.</td></tr>
<tr><td>SenderName</td><td>Body</td><td>Required. A string indicating the name that appears as the “from” in the user’s mail client (e.g. Keyfactor Command). This value is used for both configurations of RelayAuthentic-</td></tr>
</table>

Name	In	Description
		ationType.
UseSSL	Body	A Boolean indicating that mail should be delivered over TLS/SSL. Not all mail servers support this.

Table 732: POST SMTP Test Response Data

Name	Description
Host	A string indicating the fully qualified domain name of your SMTP host (e.g. corpexch02.keyexample.com).
Id	An integer indicating the Keyfactor Command reference ID of the SMTP record. This will be 1 in most environments.
Port	An integer indicating the SMTP port (e.g. 25).
RelayAuthenticationType	An integer indicating the type of authentication used to connect to the mail server. Possible values are: **Value** / **Description** 0 / Anonymous 2 / Explicit Credentials
RelayUsername	A string indicating the username of the account providing authentication to the mail server if *RelayAuthenticationType* is set to 2. The username should be provided in DOMAIN\\username format. For most mail server configurations, the username provided must have as a valid email address the email address you set in the *SenderAccount* parameter.
SenderAccount	A string indicating the sender for email messages delivered from Keyfactor Command, in the form of an email address (e.g. jsmith@keyexample.com). Depending on the email configuration in your environment, the sender account may need to be a valid user on your mail server or you may be able to put anything in this field.
SenderName	A string indicating the name that appears as the “from” in the user’s mail client (e.g. Keyfactor Command). This value is used for both configurations of *RelayAuthenticationType*.
UseSSL	A Boolean indicating that mail should be delivered over TLS/SSL. Not all mail servers support this.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.35.3 POST SMTP Test

The POST /SMTP/Test method is used to test the SMTP settings by sending a test email message. This method returns HTTP 200 OK on a success with details about the SMTP profile.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/system_setting/modify/

Table 733: POST SMTP Test Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Host</td><td>Body</td><td>Required. A string indicating the fully qualified domain name of your SMTP host (e.g. corpexch02.keyexample.com).</td></tr>
<tr><td>Id</td><td>Body</td><td>An integer indicating the Keyfactor Command reference ID of the SMTP record. This will be 1 in most environments.</td></tr>
<tr><td>Port</td><td>Body</td><td>Required. An integer indicating the SMTP port (e.g. 25).</td></tr>
<tr><td>RelayAuthenticationType</td><td>Body</td><td>An integer indicating the type of authentication used to connect to the mail server. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Anonymous</td></tr><tr><td>2</td><td>Explicit Credentials</td></tr></table></td></tr>
<tr><td>RelayPassword</td><td>Body</td><td>Required*. A string indicating the password of the user specified by RelayUsername if RelayAuthenticationType is set to 2. This field is required if RelayAuthenticationType is set to 2.
No data is output in this field on a GET.</td></tr>
<tr><td>RelayUsername</td><td>Body</td><td>Required*. A string indicating the username of the account providing authentication to the mail server if RelayAuthenticationType is set to 2. The username should be provided in DOMAIN\\username format. This field is required if RelayAuthenticationType is set to 2.
For most mail server configurations, the username provided must have as a valid email address the email address you set in the SenderAccount parameter.</td></tr>
<tr><td>SenderAccount</td><td>Body</td><td>Required. A string indicating the sender for email messages delivered from Keyfactor Command, in the form of an email address (e.g. jsmith@keyexample.com). Depending on the email configuration in your environment, the sender account may need to be a valid user on your mail server or you may be able to put anything in this field.</td></tr>
<tr><td>SenderAddress</td><td>Body</td><td>A string indicating the sender for email messages delivered from Keyfactor Command, in the form of an email address (e.g. jsmith@keyexample.com).
This is considered deprecated and may be removed in a future</td></tr>
</table>

Name	In	Description
		release.
SenderName	Body	A string indicating the name that appears as the “from” in the user’s mail client (e.g. Keyfactor Command). This value is used for both configurations of *RelayAuthenticationType*.
TestRecipient	Body	**Required**. A string indicating the recipient name, in email format (e.g. mjones@keyexample.com), for a test message to be sent using the SMTP configuration to confirm functionality.
UseSSL	Body	A Boolean indicating that mail should be delivered over TLS/SSL. Not all mail servers support this.

Table 734: POST SMTP Test Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Host</td><td>A string indicating the fully qualified domain name of your SMTP host (e.g. corpexch02.keyexample.com).</td></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the SMTP record. This will be 1 in most environments.</td></tr>
<tr><td>Port</td><td>An integer indicating the SMTP port (e.g. 25).</td></tr>
<tr><td>RelayAuthenticationType</td><td>An integer indicating the type of authentication used to connect to the mail server. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Anonymous</td></tr><tr><td>2</td><td>Explicit Credentials</td></tr></table></td></tr>
<tr><td>RelayUsername</td><td>A string indicating the username of the account providing authentication to the mail server if RelayAuthenticationType is set to 2. The username should be provided in DOMAIN\\username format.
For most mail server configurations, the username provided must have as a valid email address the email address you set in the SenderAccount parameter.</td></tr>
<tr><td>SenderAccount</td><td>A string indicating the sender for email messages delivered from Keyfactor Command, in the form of an email address (e.g. jsmith@keyexample.com). Depending on the email configuration in your environment, the sender account may need to be a valid user on your mail server or you may be able to put anything in this field.</td></tr>
<tr><td>SenderName</td><td>A string indicating the name that appears as the “from” in the user’s mail client (e.g. Keyfactor Command). This value is used for both configurations of RelayAuthenticationType.</td></tr>
<tr><td>TestRecipient</td><td>A string indicating the recipient name, in email format (e.g. mjones@keyexample.com), for a test message to be sent using the SMTP configuration to confirm functionality.</td></tr>
<tr><td>UseSSL</td><td>A Boolean indicating that mail should be delivered over TLS/SSL. Not all mail servers support this.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily

for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36 SSL

The SSL component of the Keyfactor API includes methods necessary to programmatically create, delete, edit, and list SSL networks, network ranges, and endpoints found in an SSL scan.

Table 735: SSL Endpoints

Endpoint	Method	Description	Link
/Parts/{id}	GET	Returns detailed information about a scan job for SSL discovery or monitoring.	GET SSL Parts ID on page 2305
/Endpoints/{id}	GET	Returns the details about a single endpoint discovered during SSL scanning.	GET SSL Endpoints ID on page 2307
/NetworkRanges/{id}	DELETE	Removes all network ranges from the specified SSL network.	DELETE SSL NetworkRanges ID on page 2308
/NetworkRanges/{id}	GET	Returns network range information about the specified SSL network.	GET SSL NetworkRanges ID on page 2309
/Networks/{identifier}	GET	Returns information about the specified SSL network.	GET SSL Networks Identifier on page 2310
/	GET	Returns the results of an SSL scan based on query information.	GET SSL on page 2320
/Networks	GET	Returns information about all SSL networks in Keyfactor Command.	GET SSL Networks on page 2322
/Networks	POST	Creates a new SSL network.	POST SSL Networks on page 2333
/Networks	PUT	Updates an existing SSL network.	PUT SSL Networks on page 2347

Endpoint	Method	Description	Link
/Endpoints/{id}/History	GET	Returns a list of all the SSL scanning endpoint histories for an endpoint with the given ID.	GET SSL Endpoints ID History on page 2361
/Networks/{id}/Parts	GET	Returns the scan job information for SSL discovery or monitoring.	GET SSL Networks ID Parts on page 2366
/NetworkRanges	POST	Adds network ranges to the specified SSL network.	POST SSL NetworkRanges on page 2368
/NetworkRanges	PUT	Updates network range information on the specified SSL network.	PUT SSL NetworkRanges on page 2369
/Endpoints/ReviewStatus	PUT	Used to change the *reviewed* status for a given SSL endpoint.	PUT SSL Endpoints Review Status on page 2370
/Endpoints/MonitorStatus	PUT	Used to change the *monitoring* status for a given SSL endpoint.	PUT SSL Endpoints Monitor Status on page 2371
/Endpoints/ReviewAll	PUT	Used to change the *reviewed* status for all given SSL endpoints to true.	PUT SSL Endpoints Review All on page 2371
/Endpoints/MonitorAll	PUT	Used to change the *monitoring* status for all given SSL endpoints to true.	PUT SSL Endpoints Monitor All on page 2372
/Networks/{id}/Scan	POST	Starts an SSL discovery or monitoring scan job manually.	POST SSL Networks ID Scan on page 2373
/NetworkRanges/Validate	POST	Validates all SSL networks given.	POST SSL NetworkRanges Validate on page 2374
/Networks/{id}	DELETE	Removes an SSL network from Keyfactor Command.	DELETE SSL Networks ID on page 2375

3.6.36.1 GET SSL Parts ID

The GET /SSL/Parts/{id} method retrieves information for a specific job scan segment (see GET SSL Networks ID Parts on page 2366). This method returns HTTP 200 OK on a success with details about the specified scan job segment.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 736: GET SSL Parts {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference GUID for the SSL scan job segment to be retrieved. Use the *GET /SSL/Networks/{id}/Parts* method (see GET SSL Networks ID Parts on page 2366) to retrieve a list of all the scan job segments in an SSL network to determine the SSL scan job segment's GUID.

Table 737: GET SSL Parts {id} Response Data

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td>ScanJobPartId</td><td>A string indicating the Keyfactor Command reference GUID for the scan job segment.</td></tr>
<tr><td>LogicalScanJobId</td><td>A string indicating the Keyfactor Command reference GUID for the scan job as a whole.</td></tr>
<tr><td>AgentJobId</td><td>A string indicating the Keyfactor Command reference GUID for the orchestrator that ran the job segment, if applicable. If the segment has not yet started scanning, this will show all zeros.</td></tr>
<tr><td>EstimatedEndpointCount</td><td>An integer indicating the number of endpoints that will be scanned for the segment estimated in preparation for scanning.
The number of endpoints per segment is configurable (see the SSL Maximum Scan Job Size setting on the agents tab in Application Settings: Agents Tab on page 614 in the Keyfactor Command Reference Guide).</td></tr>
<tr><td>Status</td><td>An integer indicating the status of the scan job segment. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Not Started</td></tr><tr><td>2</td><td>In Progress</td></tr><tr><td>3</td><td>Complete</td></tr></table></td></tr>
<tr><td>StatTotalEndpointCount</td><td>An integer indicating the number of endpoints that were be scanned for the segment. This value will be null if the scan is not yet complete.</td></tr>
<tr><td>StatTimedOutConnectingCount</td><td>An integer indicating the number of endpoints that timed out while attempting connections. This value will be null if the scan is not yet complete.</td></tr>
<tr><td>StatConnectionRefusedCount</td><td>An integer indicating the number of endpoints that received a connection refused while attempting connections. This value will be null if the scan is not yet complete.</td></tr>
<tr><td>StatTimedOutDownloadingCount</td><td>An integer indicating the number of endpoints that timed out while downloading while attempting connections. This value will be null if the scan is not yet complete.</td></tr>
</table>

Parameter Name	Description
StatExceptionDownloadingCount	An integer indicating the number of endpoints that encountered an exception while attempting connections. This value will be null if the scan is not yet complete.
StatNotSslCount	An integer indicating the number of endpoints that made a connection and were considered not SSL (connection on a non-SSL port such as 22 or 636). This value will be null if the scan is not yet complete.
StatBadSslHandshakeCount	An integer indicating the number of endpoints that had a bad handshake while attempting connections. This value will be null if the scan is not yet complete.
StatCertificateFoundCount	An integer indicating the number of endpoints where a certificate was found. This value will be null if the scan is not yet complete.
StatNoCertificateCount	An integer indicating the number of endpoints where the handshake got to the part of the TLS where a certificate should be returned, but did not find a certificate. This is an uncommon occurrence, so will usually be zero.
ScanJobPartsDefinitions	This is no longer in use and will always return “null”.
StartTime	A string indicating the date and time at which the scan job segment started in UTC. For jobs that have not yet started, this value will be null.
EndTime	A string indicating the date and time at which the scan job segment finished in UTC. For jobs that have not yet started, this value will be null.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.2 GET SSL Endpoints ID

The GET /SSL/Endpoints/{id} method is used to retrieve information about an endpoint found in an SSL discover or monitor scan using the EndpointId. This method returns HTTP 200 OK on a success with details of the SSL endpoints.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 738: GET SSL Endpoints {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID for the SSL endpoint to be retrieved. Use the *GET /SSL* method (see GET SSL on page 2320) to retrieve a list of all the SSL endpoints to determine the SSL endpoint's GUID.

Table 739: GET SSL Endpoints {id} Response Data

Name	Description
EndpointId	A string indicating the Keyfactor Command reference GUID for the endpoint.
NetworkId	A string indicating the Keyfactor Command reference GUID for the SSL network that scanned the endpoint.
LastHistoryId	A string indicating the Keyfactor Command reference GUID for the last history entry on the endpoint.
IpAddressBytes	A string indicating the IP address for the endpoint as bytes.
Port	An integer indicating the port on which this endpoint was found.
SNIName	A string indicating the server name indication (SNI) of the endpoint, if found.
EnableMonitor	A Boolean indicating whether monitoring is enabled on this endpoint (true) or not (false).
Reviewed	A Boolean indicating whether the endpoint has been reviewed (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.3 DELETE SSL NetworkRanges ID

The DELETE /SSL/NetworkRanges/{id} method is used to delete all the network ranges for an SSL network with the specified GUID from Keyfactor Command. This endpoint returns 204 with no

content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Tip: To delete some but not all of the network ranges for a network, use the *PUT /SSL/Networks* method to update the network and submit the request with only those network ranges you wish to retain (see PUT SSL Networks on page 2347).

Table 740: DELETE SSL Network Ranges {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference GUID for the SSL network for which to delete network ranges. Use the *GET /SSL/Networks* method (see GET SSL Networks on page 2322) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.4 GET SSL NetworkRanges ID

The GET /SSL/NetworkRanges/{id} method is used to retrieve the network ranges for an SSL network with the specified GUID from Keyfactor Command. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 741: GET SSL Network Ranges {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference GUID for the SSL network for which to retrieve network ranges. Use the *GET /SSL/Networks* method (see GET SSL Networks on page 2322) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Table 742: GET SSL Network Ranges {id} Response Data

Name	Description
ItemType	An integer indicating the type of network range. Possible values are: **Value** / **Description** 0 / Unknown 1 / IP Address 2 / Host Name 3 / Network Notation
Value	A string indicating the value for the network range, including the IP address, network notation or host name followed by the port or ports for scanning (e.g. 192.168.12.0/24:443).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.5 GET SSL Networks Identifier

The GET /SSL/Networks/{identifier} method is used to retrieve a defined SSL network according to the provided name from Keyfactor Command. This method returns HTTP 200 OK on a success with details about the SSL network.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 743: GET SSL Networks {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference GUID for the SSL network to be retrieved. Use the *GET /SSL/Networks* method (see GET SSL Networks on page 2322) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Table 744: GET SSL Networks {id} Response Data

Name	Description
NetworkId	A string indicating the Keyfactor Command reference GUID for the SSL network. This GUID is automatically set by Keyfactor Command.
Name	A string indicating the name for the SSL network.
AgentPoolName	A string indicating the name of the orchestrator pool assigned to the SSL network. See Orchestrator Pools Definition on page 470 for more information.
AgentPoolId	A string indicating the Keyfactor Command reference GUID for the orchestrator pool assigned to the SSL network.
Description	A string indicating the description of the SSL network.
Enabled	A Boolean that indicates whether scanning is enabled for the SSL network (true) or not (false). If this is set to false, no new network scans will be scheduled but any current scan will finish if one was in progress when the status was changed from true to false.
DiscoverSchedule	An object providing the discovery schedule for the SSL network group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, exactly once at 11:45 am:

<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>

Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>MonitorSchedule</td><td>An object providing the monitoring schedule for the SSL network group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).

Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>DiscoverPercentComplete</td><td>An integer indicating the percentage complete for a discovery job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
<tr><td>MonitorPercentComplete</td><td>An integer indicating the percentage complete for a monitoring job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>DiscoverStatus</td><td>An integer indicating the status of the discovery job. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Not Scheduled</td></tr><tr><td>2</td><td>Running</td></tr><tr><td>3</td><td>Previously Scanned</td></tr><tr><td>4</td><td>Scheduled</td></tr><tr><td>5</td><td>Disabled</td></tr><tr><td>6</td><td>In Quiet Hours</td></tr></table></td></tr>
<tr><td>MonitorStatus</td><td>An integer indicating the status of the monitoring job. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Not Scheduled</td></tr><tr><td>2</td><td>Running</td></tr><tr><td>3</td><td>Previously Scanned</td></tr><tr><td>4</td><td>Scheduled</td></tr><tr><td>5</td><td>Disabled</td></tr><tr><td>6</td><td>In Quiet Hours</td></tr></table></td></tr>
<tr><td>Discov-erLastScanned</td><td>A string indicating the date and time, in UTC, of the most recent discovery job. This field is populated as soon as the job is initiated and updated when the job completes.</td></tr>
<tr><td>MonitorLastScanned</td><td>A string indicating the date and time, in UTC, of the most recent monitoring job. This field is populated as soon as the job is initiated and updated when the job completes.</td></tr>
<tr><td>SslAlertRecipients</td><td>An array of strings providing the list of recipients who will receive email messages regarding the status of SSL discovery and monitoring jobs.</td></tr>
</table>

Name	Description
	Note: To improve performance in requests, data is not returned in this field for the *GET /SSL/Networks* method. Use the *GET /SSL/Networks/{id}* method to return data in this field.
AutoMonitor	A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).
GetRobots	A Boolean that indicates whether orchestrators should perform a GET /robots.txt request during scans in order to behave like a webcrawler and provide an explanation of network activity (true) or not (false).
DiscoverTimeoutMs	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however they will also increase the chance of missing a certificate on a slow or congested network.
MonitorTimeoutMs	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
ExpirationAlertDays	An integer that indicates the number of days within which to begin providing warnings regarding upcoming expiration in notification email messages.
DiscoverJobParts	An integer that indicates the number of job parts that have been created for a discovery job.
MonitorJobParts	An integer that indicates the number of job parts that have been created for a monitoring job.
QuietHours	An array of objects providing the list of scheduled quiet hour periods.
BlackoutStart	An object providing the start day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.
BlackoutEnd	An object providing the ending day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.6 GET SSL

The GET /SSL method is used to return a list of all discovered SSL endpoints, limited by the provided parameters. This method returns HTTP 200 OK on a success with details about the requested endpoints.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 745: GET SSL Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns. For querying guidelines, refer to: Using the Discovery Results Search Feature on page 474. The query fields supported for this endpoint are: • *AgentPoolName* • *CertificateCN* • *CertificateFound* (True, False) • *Status* (6-Certificate Found, 1-Timed Out Connecting, 2-Exception Connecting, 3-Timed Out Downloading, 4-Exception Downloading, 5-Not SSL, 7-Exception in Sql, 8-Invalid or Unreachable Host, 9-Connection Refused, 10-Bad SSL Handshake, 11-Client Authentication Failed, 12-No Certificate, 13-SSL Refused, 14-Not Probed, 0-Unknown) • *IpAddress* • *IsMonitored* (True, False) • *IssuerDN* • *NetworkName* • *Port* • *ReverseDNS* • *Reviewed* (True, False) • *SelfSigned* (True, False) • *SNIName*
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *ReverseDNS*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 746: GET SSL Response Data

Name	Description
EndpointId	A string indicating the Keyfactor Command reference GUID for the endpoint.
ReverseDNS	A string indicating the DNS name resolved for the endpoint based on the discovered IP address. If a host name could not be resolved, this will be the IP address.
SNIName	A string indicating the server name indication (SNI) of the endpoint, if found.
IpAddress	A string indicating the IP address of the endpoint.
Port	An integer indicating the port at which the endpoint was found.
CertificateFound	A Boolean indicating whether a certificate was found at the endpoint (true) or not (false).
AgentPoolName	A string indicating the name of the orchestrator pool that performed a scan (discovery or monitoring) on the endpoint.
NetworkName	A string indicating the name of the SSL network that performed a scan (discovery or monitoring) on the endpoint.
MonitorStatus	A Boolean indicating whether the endpoint should be monitored (true) or not (false).
CertificateCN	A string indicating the common name of the certificate that was found at the endpoint.
Reviewed	A Boolean indicating whether the endpoint has been reviewed (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.7 GET SSL Networks

The GET /SSL/Networks method is used to retrieve one or more SSL networks from Keyfactor Command. Results can be limited to selected SSL networks using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details about the specified SSL networks.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 747: GET SSL Networks Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Network Scan Details Search on page 467. The query fields supported for this endpoint are: • Name • Pool
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Name*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.This field is optional.

Table 748: GET SSL Networks Response Data

Name	Description
NetworkId	A string indicating the Keyfactor Command reference GUID for the SSL network. This GUID is automatically set by Keyfactor Command.
Name	A string indicating the name for the SSL network.
AgentPoolName	A string indicating the name of the orchestrator pool assigned to the SSL network. See Orchestrator Pools Definition on page 470 for more information.
AgentPoolId	A string indicating the Keyfactor Command reference GUID for the orchestrator pool assigned to the SSL network.
Description	A string indicating the description of the SSL network.
Enabled	A Boolean that indicates whether scanning is enabled for the SSL network (true) or not (false). If this is set to false, no new network scans will be scheduled but any current scan will finish if one was in progress when the status was changed from true to false.
DiscoverSchedule	An object providing the discovery schedule for the SSL network group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>MonitorSchedule</td><td>An object providing the monitoring schedule for the SSL network group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour: `"Interval": { "Minutes": 60 }`
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2023-11-25T23:30:00Z" }`
Weekly	A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}
</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre></td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>DiscovererPercentComplete</td><td>An integer indicating the percentage complete for a discovery job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
<tr><td>MonitorPercentComplete</td><td>An integer indicating the percentage complete for a monitoring job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>DiscoverStatus</td><td>An integer indicating the status of the discovery job. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Not Scheduled</td></tr><tr><td>2</td><td>Running</td></tr><tr><td>3</td><td>Previously Scanned</td></tr><tr><td>4</td><td>Scheduled</td></tr><tr><td>5</td><td>Disabled</td></tr><tr><td>6</td><td>In Quiet Hours</td></tr></table></td></tr>
<tr><td>MonitorStatus</td><td>An integer indicating the status of the monitoring job. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Not Scheduled</td></tr><tr><td>2</td><td>Running</td></tr><tr><td>3</td><td>Previously Scanned</td></tr><tr><td>4</td><td>Scheduled</td></tr><tr><td>5</td><td>Disabled</td></tr><tr><td>6</td><td>In Quiet Hours</td></tr></table></td></tr>
<tr><td>Discov-erLastScanned</td><td>A string indicating the date and time, in UTC, of the most recent discovery job. This field is populated as soon as the job is initiated and updated when the job completes.</td></tr>
<tr><td>MonitorLastScanned</td><td>A string indicating the date and time, in UTC, of the most recent monitoring job. This field is populated as soon as the job is initiated and updated when the job completes.</td></tr>
<tr><td>SslAlertRecipients</td><td>An array of strings providing the list of recipients who will receive email messages regarding the status of SSL discovery and monitoring jobs.</td></tr>
</table>

Name	Description
	Note: To improve performance in requests, data is not returned in this field for the *GET /SSL/Networks* method. Use the *GET /SSL/Networks/{id}* method to return data in this field.
GetRobots	A Boolean that indicates whether orchestrators should perform a GET /robots.txt request during scans in order to behave like a webcrawler and provide an explanation of network activity (true) or not (false).
DiscoverTimeoutMs	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however they will also increase the chance of missing a certificate on a slow or congested network.
MonitorTimeoutMs	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
ExpirationAlertDays	An integer that indicates the number of days within which to begin providing warnings regarding upcoming expiration in notification email messages.
DiscoverJobParts	An integer that indicates the number of job parts that have been created for a discovery job.
MonitorJobParts	An integer that indicates the number of job parts that have been created for a monitoring job.
QuietHours	An array of objects providing the list of scheduled quiet hour periods.
BlackoutStart	An object providing the start day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.
BlackoutEnd	An object providing the ending day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.8 POST SSL Networks

The POST /SSL/Networks method is used to create an SSL network in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the new SSL network.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Table 749: POST SSL Networks Input Parameters

Name	In	Description
NetworkId	Body	A string indicating the Keyfactor Command reference GUID for the SSL network. This GUID is automatically set by Keyfactor Command.
Name	Body	**Required**. A string indicating the name for the SSL network.
AgentPoolName	Body	**Required**. A string indicating the name of the orchestrator pool assigned to the SSL network. See for more information.
AgentPoolId	Body	A string indicating the Keyfactor Command reference GUID for the orchestrator pool assigned to the SSL network.
Description	Body	**Required**. A string indicating the description of the SSL network.
Enabled	Body	A Boolean that indicates whether scanning is enabled for the SSL network (true) or not (false). If this is set to false, no new network scans will be scheduled but any current scan will finish if one was in progress when the status was changed from true to false.
DiscoverSchedule	Body	An object providing the discovery schedule for the SSL network group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>}</pre></td></tr>
<tr><td>ExactlyO-nce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>MonitorSchedule</td><td>Bod-y</td><td>An object providing the monitoring schedule for the SSL network group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run imme-diately (true) or not (false).
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.
</td></tr>
</table>
</td></tr>
<tr><td>DiscovererPercentComplete</td><td>Body</td><td>An integer indicating the percentage complete for a discovery job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>This field is for reference and is not configurable.</td></tr>
<tr><td>MonitorPercentComplete</td><td>Body</td><td>An integer indicating the percentage complete for a monitoring job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins. This field is for reference and is not configurable.</td></tr>
<tr><td>DiscoverStatus</td><td>Body</td><td>An integer indicating the status of the discovery job. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Not Scheduled</td></tr><tr><td>2</td><td>Running</td></tr><tr><td>3</td><td>Previously Scanned</td></tr><tr><td>4</td><td>Scheduled</td></tr><tr><td>5</td><td>Disabled</td></tr><tr><td>6</td><td>In Quiet Hours</td></tr></table></td></tr>
<tr><td>MonitorStatus</td><td>Body</td><td>An integer indicating the status of the monitoring job. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Not Scheduled</td></tr><tr><td>2</td><td>Running</td></tr><tr><td>3</td><td>Previously Scanned</td></tr><tr><td>4</td><td>Scheduled</td></tr><tr><td>5</td><td>Disabled</td></tr><tr><td>6</td><td>In Quiet Hours</td></tr></table></td></tr>
<tr><td>Discov-</td><td>Bod-</td><td>A string indicating the date and time, in UTC, of the most recent discovery</td></tr>
</table>

Name	In	Description
erLastScanned	y	job. This field is populated as soon as the job is initiated and updated when the job completes. This field is for reference and is not configurable.
MonitorLastScanned	Body	A string indicating the date and time, in UTC, of the most recent monitoring job. This field is populated as soon as the job is initiated and updated when the job completes. This field is for reference and is not configurable.
SslAlertRecipients	Body	An array of strings providing the list of recipients who will receive email messages regarding the status of SSL discovery and monitoring jobs. **Note:** To improve performance in requests, data is not returned in this field for the *GET /SSL/Networks* method. Use the *GET /SSL/Networks/{id}* method to return data in this field.
GetRobots	Body	A Boolean that indicates whether orchestrators should perform a GET /robots.txt request during scans in order to behave like a webcrawler and provide an explanation of network activity (true) or not (false).
DiscoverTimeoutMs	Body	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however they will also increase the chance of missing a certificate on a slow or congested network.
MonitorTimeoutMs	Body	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
ExpirationAlertDays	Body	An integer that indicates the number of days within which to begin providing warnings regarding upcoming expiration in notification email messages.
DiscoverJobParts	Body	An integer that indicates the number of job parts that have been created for a discovery job. This field is for reference and is not configurable.
MonitorJobParts	Body	An integer that indicates the number of job parts that have been created for a monitoring job. This field is for reference and is not configurable.

QuietHours	Body	An array of objects providing the list of scheduled quiet hour periods. For example:
		
		`"QuietHours": [`
		`  {`
		`  "StartDay": "Monday",`

Name	In	Description

		<pre>"StartTime": "2022-11-21T14:00:08Z",
		"EndDay": "Tuesday",
		"EndTime": "2022-11-22T14:00:08Z"
		},
		{
		"StartDay": "Saturday",
		"StartTime": "2022-11-26T04:00:08Z",
		"EndDay": "Sunday",
		"EndTime": "2022-11-27T16:00:08Z"
		}
		]</pre>

BlackoutStart	Body	An object providing the start day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.
BlackoutEnd	Body	An object providing the ending day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.

Table 750: POST SSL Networks Response Data

Name	Description
NetworkId	A string indicating the Keyfactor Command reference GUID for the SSL network. This GUID is automatically set by Keyfactor Command.
Name	A string indicating the name for the SSL network.
AgentPoolName	A string indicating the name of the orchestrator pool assigned to the SSL network. See Orchestrator Pools Definition on page 470 for more information.
AgentPoolId	A string indicating the Keyfactor Command reference GUID for the orchestrator pool assigned to the SSL network.
Description	A string indicating the description of the SSL network.
Enabled	A Boolean that indicates whether scanning is enabled for the SSL network (true) or not (false). If this is set to false, no new network scans will be scheduled but any current scan will finish if one was in progress when the status was changed from true to false.
DiscoverSchedule	An object providing the discovery schedule for the SSL network group.
MonitorSchedule	An object providing the monitoring schedule for the SSL network group.
DiscoverPercentComplete	An integer indicating the percentage complete for a discovery job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.
MonitorPercentComplete	An integer indicating the percentage complete for a monitoring job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.
DiscoverStatus	An integer indicating the status of the discovery job. Possible values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Unknown</td></tr>
<tr><td>1</td><td>Not Scheduled</td></tr>
<tr><td>2</td><td>Running</td></tr>
<tr><td>3</td><td>Previously Scanned</td></tr>
<tr><td>4</td><td>Scheduled</td></tr>
<tr><td>5</td><td>Disabled</td></tr>
<tr><td>6</td><td>In Quiet Hours</td></tr>
</table>
</td></tr>
<tr><td>MonitorStatus</td><td>An integer indicating the status of the monitoring job. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Unknown</td></tr>
<tr><td>1</td><td>Not Scheduled</td></tr>
<tr><td>2</td><td>Running</td></tr>
<tr><td>3</td><td>Previously Scanned</td></tr>
<tr><td>4</td><td>Scheduled</td></tr>
<tr><td>5</td><td>Disabled</td></tr>
<tr><td>6</td><td>In Quiet Hours</td></tr>
</table>
</td></tr>
<tr><td>DiscoverLastScanned</td><td>A string indicating the date and time, in UTC, of the most recent discovery job. This field is populated as soon as the job is initiated and updated when the job completes.</td></tr>
<tr><td>MonitorLastScanned</td><td>A string indicating the date and time, in UTC, of the most recent monitoring job. This field is populated as soon as the job is initiated and updated when the job completes.</td></tr>
<tr><td>SslAlertRecipients</td><td>An array of strings providing the list of recipients who will receive email messages regarding the status of SSL discovery and monitoring jobs.</td></tr>
</table>

Name	Description
	Note: To improve performance in requests, data is not returned in this field for the *GET /SSL/Networks* method. Use the *GET /SSL/Networks/{id}* method to return data in this field.
AutoMonitor	A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).
GetRobots	A Boolean that indicates whether orchestrators should perform a GET /robots.txt request during scans in order to behave like a webcrawler and provide an explanation of network activity (true) or not (false).
DiscoverTimeoutMs	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however they will also increase the chance of missing a certificate on a slow or congested network.
MonitorTimeoutMs	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
ExpirationAlertDays	An integer that indicates the number of days within which to begin providing warnings regarding upcoming expiration in notification email messages.
DiscoverJobParts	An integer that indicates the number of job parts that have been created for a discovery job.
MonitorJobParts	An integer that indicates the number of job parts that have been created for a monitoring job.
QuietHours	An array of objects providing the list of scheduled quiet hour periods.
BlackoutStart	An object providing the start day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.
BlackoutEnd	An object providing the ending day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.9 PUT SSL Networks

The PUT /SSL/Networks method is used to update an SSL network in Keyfactor Command. This method returns HTTP 200 OK on a success with details for the SSL network.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 751: PUT SSL Networks Input Parameters

Name	In	Description
NetworkId	Body	A string indicating the Keyfactor Command reference GUID for the SSL network. This GUID is automatically set by Keyfactor Command.
Name	Body	**Required**. A string indicating the name for the SSL network.
AgentPoolName	Body	**Required**. A string indicating the name of the orchestrator pool assigned to the SSL network. See for more information.
AgentPoolId	Body	A string indicating the Keyfactor Command reference GUID for the orchestrator pool assigned to the SSL network.
Description	Body	**Required**. A string indicating the description of the SSL network.
Enabled	Body	A Boolean that indicates whether scanning is enabled for the SSL network (true) or not (false). If this is set to false, no new network scans will be scheduled but any current scan will finish if one was in progress when the status was changed from true to false.
DiscoverSchedule	Body	An object providing the discovery schedule for the SSL network group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:

Name	Description
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

Name	Description
Minutes	An integer indicating the number of minutes between each interval.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

```
}
```

</td></tr>
<tr><td>ExactlyO-nce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:

```
"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}
```

Tip: In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *ExactlyOnce*.</td></tr>
</table>
</td></tr>
<tr><td>MonitorSchedule</td><td>Bod-y</td><td>An object providing the monitoring schedule for the SSL network group. The schedule can be off (unset) or one of the supported values. Supported schedule values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).

Tip: In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:

<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2023-11-25T23:30:00Z"
}</pre></td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. “Sunday”).</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>Monthly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Day</td><td>The number of the day, in the month, to run the job.</td></tr>
</table>
For example, on the first of every month at 5:30 pm:
<pre>"Monthly": {
   "Day": 1
   "Time": "2023-11-27T17:30:00Z"
}</pre>
</td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.
</td></tr>
</table>
</td></tr>
<tr><td>DiscovererPercentComplete</td><td>Body</td><td>An integer indicating the percentage complete for a discovery job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>This field is for reference and is not configurable.</td></tr>
<tr><td>MonitorPercentComplete</td><td>Body</td><td>An integer indicating the percentage complete for a monitoring job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins. This field is for reference and is not configurable.</td></tr>
<tr><td>DiscoverStatus</td><td>Body</td><td>An integer indicating the status of the discovery job. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Not Scheduled</td></tr><tr><td>2</td><td>Running</td></tr><tr><td>3</td><td>Previously Scanned</td></tr><tr><td>4</td><td>Scheduled</td></tr><tr><td>5</td><td>Disabled</td></tr><tr><td>6</td><td>In Quiet Hours</td></tr></table></td></tr>
<tr><td>MonitorStatus</td><td>Body</td><td>An integer indicating the status of the monitoring job. Possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Not Scheduled</td></tr><tr><td>2</td><td>Running</td></tr><tr><td>3</td><td>Previously Scanned</td></tr><tr><td>4</td><td>Scheduled</td></tr><tr><td>5</td><td>Disabled</td></tr><tr><td>6</td><td>In Quiet Hours</td></tr></table></td></tr>
<tr><td>Discov-</td><td>Bod-</td><td>A string indicating the date and time, in UTC, of the most recent discovery</td></tr>
</table>

Name	In	Description
erLastScanned	y	job. This field is populated as soon as the job is initiated and updated when the job completes. This field is for reference and is not configurable.
MonitorLastScanned	Body	A string indicating the date and time, in UTC, of the most recent monitoring job. This field is populated as soon as the job is initiated and updated when the job completes. This field is for reference and is not configurable.
SslAlertRecipients	Body	An array of strings providing the list of recipients who will receive email messages regarding the status of SSL discovery and monitoring jobs. **Note:** To improve performance in requests, data is not returned in this field for the *GET /SSL/Networks* method. Use the *GET /SSL/Networks/{id}* method to return data in this field.
GetRobots	Body	A Boolean that indicates whether orchestrators should perform a GET /robots.txt request during scans in order to behave like a webcrawler and provide an explanation of network activity (true) or not (false).
DiscoverTimeoutMs	Body	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however they will also increase the chance of missing a certificate on a slow or congested network.
MonitorTimeoutMs	Body	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
ExpirationAlertDays	Body	An integer that indicates the number of days within which to begin providing warnings regarding upcoming expiration in notification email messages.
DiscoverJobParts	Body	An integer that indicates the number of job parts that have been created for a discovery job. This field is for reference and is not configurable.
MonitorJobParts	Body	An integer that indicates the number of job parts that have been created for a monitoring job. This field is for reference and is not configurable.

QuietHours	Body	An array of objects providing the list of scheduled quiet hour periods. For example:
		
		`"QuietHours": [`
		`  {`
		`  "StartDay": "Monday",`

Name	In	Description

		<pre>"StartTime": "2022-11-21T14:00:08Z",
		"EndDay": "Tuesday",
		"EndTime": "2022-11-22T14:00:08Z"
		},
		{
		"StartDay": "Saturday",
		"StartTime": "2022-11-26T04:00:08Z",
		"EndDay": "Sunday",
		"EndTime": "2022-11-27T16:00:08Z"
		}
		]</pre>

BlackoutStart	Body	An object providing the start day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.
BlackoutEnd	Body	An object providing the ending day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.

Table 752: PUT SSL Networks Response Data

Name	Description
NetworkId	A string indicating the Keyfactor Command reference GUID for the SSL network. This GUID is automatically set by Keyfactor Command.
Name	A string indicating the name for the SSL network.
AgentPoolName	A string indicating the name of the orchestrator pool assigned to the SSL network. See Orchestrator Pools Definition on page 470 for more information.
AgentPoolId	A string indicating the Keyfactor Command reference GUID for the orchestrator pool assigned to the SSL network.
Description	A string indicating the description of the SSL network.
Enabled	A Boolean that indicates whether scanning is enabled for the SSL network (true) or not (false). If this is set to false, no new network scans will be scheduled but any current scan will finish if one was in progress when the status was changed from true to false.
DiscoverSchedule	An object providing the discovery schedule for the SSL network group.
MonitorSchedule	An object providing the monitoring schedule for the SSL network group.
DiscoverPercentComplete	An integer indicating the percentage complete for a discovery job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.
MonitorPercentComplete	An integer indicating the percentage complete for a monitoring job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.
DiscoverStatus	An integer indicating the status of the discovery job. Possible values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Unknown</td></tr>
<tr><td>1</td><td>Not Scheduled</td></tr>
<tr><td>2</td><td>Running</td></tr>
<tr><td>3</td><td>Previously Scanned</td></tr>
<tr><td>4</td><td>Scheduled</td></tr>
<tr><td>5</td><td>Disabled</td></tr>
<tr><td>6</td><td>In Quiet Hours</td></tr>
</table>
</td></tr>
<tr><td>MonitorStatus</td><td>An integer indicating the status of the monitoring job. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Unknown</td></tr>
<tr><td>1</td><td>Not Scheduled</td></tr>
<tr><td>2</td><td>Running</td></tr>
<tr><td>3</td><td>Previously Scanned</td></tr>
<tr><td>4</td><td>Scheduled</td></tr>
<tr><td>5</td><td>Disabled</td></tr>
<tr><td>6</td><td>In Quiet Hours</td></tr>
</table>
</td></tr>
<tr><td>DiscoverLastScanned</td><td>A string indicating the date and time, in UTC, of the most recent discovery job. This field is populated as soon as the job is initiated and updated when the job completes.</td></tr>
<tr><td>MonitorLastScanned</td><td>A string indicating the date and time, in UTC, of the most recent monitoring job. This field is populated as soon as the job is initiated and updated when the job completes.</td></tr>
<tr><td>SslAlertRecipients</td><td>An array of strings providing the list of recipients who will receive email messages regarding the status of SSL discovery and monitoring jobs.</td></tr>
</table>

Name	Description
	Note: To improve performance in requests, data is not returned in this field for the *GET /SSL/Networks* method. Use the *GET /SSL/Networks/{id}* method to return data in this field.
AutoMonitor	A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).
GetRobots	A Boolean that indicates whether orchestrators should perform a GET /robots.txt request during scans in order to behave like a webcrawler and provide an explanation of network activity (true) or not (false).
DiscoverTimeoutMs	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however they will also increase the chance of missing a certificate on a slow or congested network.
MonitorTimeoutMs	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
ExpirationAlertDays	An integer that indicates the number of days within which to begin providing warnings regarding upcoming expiration in notification email messages.
DiscoverJobParts	An integer that indicates the number of job parts that have been created for a discovery job.
MonitorJobParts	An integer that indicates the number of job parts that have been created for a monitoring job.
QuietHours	An array of objects providing the list of scheduled quiet hour periods.
BlackoutStart	An object providing the start day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.
BlackoutEnd	An object providing the ending day and time for the first scheduled quiet hours period. See *QuietHours* for the full list of scheduled quiet hours. If both a Blackout period and a QuietHours period are configured for the same network, the QuietHours period will be used. This is considered deprecated and may be removed in a future release.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.10 GET SSL Endpoints ID History

The GET /SSL/Endpoints/{id}/History method is used to return a list of history found for a given SSL endpoint. URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details for the specified endpoint.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 753: GET SSL Endpoints {id} History Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference GUID for the SSL endpoint for which to return history information. Use the *GET /SSL* method (see GET SSL on page 2320) to retrieve a list of all the SSL endpoints to determine the GUID of the desired endpoint.
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.

Table 754: GET SSL Endpoints {id} History Response Data

Name	Description
HistoryId	A string indicating the Keyfactor Command reference GUID for the history entry.
EndpointId	A string indicating the Keyfactor Command reference GUID for the endpoint with which the history is associated.
AuditId	An integer indicating the Keyfactor Command ID used to track progress during scan jobs.
Timestamp	A string indicating the date and time the history entry was created.
Status	An integer containing the status of the scan for which the history item was created. The possible values are: **Value** / **Description** 0 – Unknown 1 – TimeOutConnecting 2 – ExceptionConnecting 3 – TimeoutDownloading 4 – ExceptionDownloading 5 – NotSsl 6 – CertificateFound 7 – ExceptionInSql 8 – InvalidOrUnreachableHost 9 – ConnectionRefused 10 – BadSslHandshake 11 – ClientAuthenticationFailed 12 – NoCertificate 13 – SslRefused 14 – NotProbed
JobType	An integer containing the type of scan job from which the history entry was created.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>The possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Unknown</td></tr><tr><td>1</td><td>Discovery</td></tr><tr><td>2</td><td>Monitoring</td></tr><tr><td>3</td><td>Compliance</td></tr></table></td></tr>
<tr><td>ProbeType</td><td>An integer containing the type of connection made to the endpoint for the scan from which the history entry was created. The possible values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>2</td><td>SSLv2</td></tr><tr><td>3</td><td>TLS</td></tr></table></td></tr>
<tr><td>ReverseDNS</td><td>A string indicating the DNS name of the endpoint resolved based on the discovered IP address at the time the history entry was created. If a host name could not be resolved, this will be the IP address.</td></tr>
<tr><td>HistoryCertificates</td><td>An array of objects indicating the certificates found at the endpoint during the scan from which the history entry was created. Information includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the certificate.</td></tr><tr><td>IssuedDN</td><td>A string indicating the distinguished name of the certificate.</td></tr><tr><td>SerialNumber</td><td>A string indicating the serial number of the certificate.</td></tr><tr><td>NotBefore</td><td>A string indicating the date, in UTC, on which the certificate was issued by the certificate authority.</td></tr><tr><td>NotAfter</td><td>A string indicating the date, in UTC, on which the certificate expires.</td></tr></table></td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>SigningAlgorithm</td><td>A string indicating the algorithm used to sign the certificate.</td></tr><tr><td>Thumbprint</td><td>A string indicating the thumbprint of the certificate.</td></tr><tr><td>IssuerDN</td><td>A string indicating the distinguished name of the issuer.</td></tr><tr><td>IssuedCN</td><td>A string indicating the common name of the certificate.</td></tr></table>

Name	Description

Name	Description
SubjectAltNameElements	

Name	Description
Id	An integer containing the Keyfactor Command reference ID of the SAN Element.
Value	A string indicating the value of the SAN Element.
Type	An integer containing the type of SAN element. The possible values are:

Value	Description
0	Other Name
1	RFC 822 Name
2	DNS Name
3	X400 Address
4	Directory Name
5	Ediparty Name
6	Uniform Resource Identifier
7	IP Address
8	Registered Id
100	MS_NTPrincipalName
101	MS_NTDSReplication
999	Unknown

ValueHash	A string indicating a hash of the SAN value.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.11 GET SSL Networks ID Parts

The GET /SSL/Networks/{id}/Parts method returns a list of scan job segments for an SSL network defined in Keyfactor Command. This method returns HTTP 200 OK on a success with the scan job segments for the specified SSL network. The results will only include more than one segment if the SSL management job was broken up into segments due to the number of endpoints it contained. The number of endpoints per segment is configurable (see the *SSL Maximum Discovery Scan Job Size* and *SSL Maximum Monitoring Scan Job Size* settings in Application Settings: Agents Tab on page 614. The results from this method are of the currently in progress job or the latest completed job.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 755: GET SSL Networks {id} Parts Input Parameters

Name	In	Description
ID	Path	**Required**. The Keyfactor Command reference GUID for the SSL network for which to retrieve scan job segments. Use the *GET /SSL/Networks* method (see GET SSL Networks on page 2322) to retrieve a list of all the SSL networks to determine the SSL network's GUID.
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Network Scan Details Search on page 467. The query fields supported for this endpoint are: • Agent • EndTime • EndpointCount • Status • StartTime
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Status*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 756: GET SSL Networks {id} Parts Response Data

Name	Description
ScanJobPartId	A string indicating the Keyfactor Command reference GUID for the scan job segment.
Agent	A string indicating the client machine name of the orchestrator that ran the scan job segment.
Status	An integer indicating the status of the scan job segment. Possible values are: **Value** / **Description** 1 / Not Started 2 / In Progress 3 / Complete
StartTime	A string indicating the date and time at which the scan job segment started in UTC. For jobs that have not yet started, this value will be null.
EndTime	A string indicating the date and time at which the scan job segment finished in UTC. For jobs that are in progress, this value will be null.
EndpointCount	An integer indicating the number of endpoints scanned for the segment.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.12 POST SSL NetworkRanges

The POST /SSL/NetworkRanges method is used to add network ranges to a specified SSL network. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Table 757: POST SSL Network Ranges Input Parameters

Name	In	Description
NetworkId	Body	**Required**. A string indicating the Keyfactor Command reference GUID for the SSL network. Use the GET /SSL/Networks method (see GET SSL Networks on page 2322) to retrieve a list of your defined SSL networks to determine the GUID of the SSL network you want to use.
Ranges	Body	**Required**. An array of strings indicating the value(s) for the network range(s), including the IP address, network notation or host name followed by the port or ports for scanning (e.g. 192.168.12.0/24:443). For example: `"Ranges": [ "192.168.12.0/24:443", "keyexample.com:443", "222.33.44.55:443" ]`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.13 PUT SSL NetworkRanges

The PUT /SSL/NetworkRanges method is used to update network ranges for a specified SSL network. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 758: PUT SSL Network Ranges {id} Input Parameters

Name	In	Description
NetworkId	Body	**Required**. A string indicating the Keyfactor Command reference GUID for the SSL network. Use the GET /SSL/Networks method (see GET SSL Networks on page 2322) to retrieve a list of your defined SSL networks to determine the GUID of the SSL network you want to use.
Ranges	Body	**Required**. An array of strings indicating the value(s) for the network range(s), including the IP address, network notation or host name followed by the port or ports for scanning (e.g. 192.168.12.0/24:443). For example: `"Ranges": [ "192.168.12.0/24:443", "keyexample.com:443", "222.33.44.55:443" ]`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.14 PUT SSL Endpoints Review Status

The PUT /SSL/Endpoints/ReviewStatus method is used to update the reviewed status of the specified endpoint. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Table 759: PUT SSL Endpoints Review Status Input Parameters

Name	In	Description
Id	Body	**Required**. A string indicating the Keyfactor Command reference GUID for the endpoint to be updated. Use the *GET /SSL* method (see GET SSL on page 2320) to retrieve a list of all the SSL endpoints to determine the GUID of the desired endpoint.
Status	Body	**Required**. A Boolean indicating whether the endpoint should be marked as reviewed (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.15 PUT SSL Endpoints Monitor Status

The PUT /SSL/Endpoints/MonitorStatus method is used to update the monitoring status of the specified endpoint. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Table 760: PUT SSL Endpoints Monitor Status Input Parameters

Name	In	Description
Id	Body	**Required**. A string indicating the Keyfactor Command reference GUID for the endpoint to be updated. Use the *GET /SSL* method (see GET SSL on page 2320) to retrieve a list of all the SSL endpoints to determine the GUID of the desired endpoint.
Status	Body	**Required**. A Boolean indicating whether monitoring should be enabled on this endpoint (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.16 PUT SSL Endpoints Review All

The PUT /SSL/Endpoints/ReviewAll method is used to update all endpoints in the given query to set the reviewed status to true. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Table 761: PUT SSL Endpoints Review All Input Parameter

Name	In	Description
Query	Query	A string containing a query to limit the endpoints that will be marked as reviewed (e.g. field1 -eq value1 AND field2 -gt value2). If this parameter is not supplied, all endpoints will be marked as reviewed. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns. For querying guidelines, refer to: Using the Discovery Results Search Feature on page 474.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.17 PUT SSL Endpoints Monitor All

The PUT /SSL/Endpoint/MonitorAll method is used to update all endpoints in the given query to set the monitoring status to true. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Table 762: PUT SSL Endpoints Monitor All Input Parameter

Name	In	Description
Query	Query	A string containing a query to limit the endpoints that will be marked as monitored (e.g. field1 -eq value1 AND field2 -gt value2). If this parameter is not supplied, all endpoints will be marked as monitored. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search dropdowns. For querying guidelines, refer to: Using the Discovery Results Search Feature on page 474.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.18 POST SSL Networks ID Scan

The POST /SSL/Networks/{id}/Scan method is used to initiate a scan job for an SSL network defined in Keyfactor Command. A scan may be manually initiated for a configured network at any time that a scan is not already running for the network or the network is not in quiet hours. When you initiate a scan, you can choose whether to run a discovery scan, a monitoring scan, or both. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Table 763: POST SSL Networks {id} Scan Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID for the SSL network for which to initiate a manual scan. Use the *GET /SSL/Networks* method (see GET SSL Networks on page 2322) to retrieve a list of all the SSL networks to determine the SSL network's GUID.
Discovery	Body	A Boolean indicating whether to initiate a manual discovery scan (true) or not (false).
Monitoring	Body	A Boolean indicating whether to initiate a manual monitoring scan (true) or not (false).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.19 POST SSL Networks ID Reset

The POST /SSL/Networks/{id}/Reset method is used to reset an SSL scan. Reset deletes all scan jobs, scan job parts, logical scan jobs, and current schedules associated with the selected network. The agent job status relating to the SSL scans is set to failed and completed, and the agent is forced to register for a new session. Afterward, *Scan Now* is enabled to allow you to initiate a manual scan. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:

/ssl/modify/

Table 764: POST SSL Networks {id} Reset Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID for the SSL network for which to reset. Use the *GET /SSL/Networks* method (see GET SSL Networks on page 2322) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.20 POST SSL NetworkRanges Validate

The POST /SSL/NetworkRanges/Validate method ensures that network ranges supplied in the request are of valid structure. This endpoint returns 204 with no content upon success. Use this method to test a proposed network range before using POST /SSL/NetworkRanges or PUT /SSL/NetworkRanges to configure it for an SSL network.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/read/

Table 765: POST SSL Network Ranges Validate Input Parameters

Name	In	Description
networkRangesToVerify	Body	**Required**. An array of strings indicating the network ranges to validate. For example: `["10.5.4.0/24:443","192.168.12.0/16:443,22","keyexample.com:443"]`

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.36.21 DELETE SSL Networks ID

The DELETE /SSL/Networks/{id} method is used to delete an SSL network with the specified GUID from Keyfactor Command. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/ssl/modify/

Table 766: DELETE SSL Networks {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the Keyfactor Command reference GUID for the SSL network to be deleted. Use the *GET /SSL/Networks* method (see GET SSL Networks on page 2322) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.37 Status

The Status component of the Keyfactor API includes methods necessary to retrieve the current list of Keyfactor API endpoints.

Table 767: Status Endpoints

Endpoint	Method	Description	Link
/Endpoints	GET	Returns a list of the Keyfactor API endpoints.	GET Status Endpoints below

3.6.37.1 GET Status Endpoints

The GET /Status/Endpoints method returns a list of all the endpoints currently available for use in the Keyfactor API. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849). This method returns HTTP 200 OK on a success with a list of all the API endpoints available in the Keyfactor API.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
None

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.38 Templates

The Templates component of the Keyfactor API includes methods necessary to programmatically edit, import and retrieve templates. Editing a template in Keyfactor Command will only apply within the software.

Table 768: Templates Endpoints

Endpoint	Method	Description	Link
/{id}	GET	Returns information about the specified template.	GET Templates ID on the next page
/Settings	GET	Returns the global template policy settings.	GET Templates Settings on page 2395
/Settings	PUT	Sets global values for template policy.	PUT Templates Settings on page 2402
/SubjectParts	GET	Returns a list of supported subject parts for template regular expressions and default subjects.	GET Templates Subject Parts on page 2421
/	GET	Returns a list of templates.	GET Templates on page 2422
/	PUT	Updates selected settings for the specified template.	PUT Templates on page 2435
/Import	POST	Import templates from a specified configuration tenant into Keyfactor Command	POST Templates Import on page 2470

3.6.38.1 GET Templates ID

The GET /Templates/{id} method is used to retrieve a specified template from Keyfactor Command. This method returns HTTP 200 OK on a success with details about the requested template.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_templates/read/

Table 769: GET Templates {id} Input Parameters

Name	In	Description
id	Path	**Required**. An integer specifying the ID of the template in Keyfactor Command. Use the *GET /Templates* method (see GET Templates on page 2422) to retrieve a list of all the templates to determine the template ID.

Table 770: GET Templates {id} Response Data

Name	Description
Id	An integer indicating the ID of the template in Keyfactor Command.
CommonName	A string containing the common name (short name) of the template. This name typically does not contain spaces. For a template created using a Microsoft management tool, this will be the Microsoft template name. For a template generated for an EJBCA CA, this will be built using a naming scheme of *<end entity profile name>_ <certificate profile name>*. This field is populated based on information retrieved from the CA and is not configurable.
TemplateName	A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name. For a template generated for an EJBCA CA, this will be built using a naming scheme of *<end entity profile name> (<certificate profile name>)*. This field is populated based on information retrieved from the CA and is not configurable.
Oid	A string containing the object ID of the template in Active Directory. For Microsoft templates, this field is populated from Active Directory. For EJBCA templates, this field is generated within Keyfactor Command as an object identifier, but does not follow official OID conventions. The field is not configurable.
KeySize	A string indicating the minimum supported key size of the template as returned by the CA. This value is calculated based on the algorithms provided in the template from the CA (see *KeyAlgorithms*). The algorithm key types and sizes are evaluated in order (RSA, ECC, Ed448, and Ed25519) and from these, the minimum type and size is determined. For example, if the template supports RSA, Ed448, and Ed25519, the minimum key type will be evaluated to RSA. Then for that algorithm, the minimum key size returned by the CA will be selected (e.g. 2048 if 2048 and 4096 are returned for RSA). See the *KeyAlgorithms* field for the complete list of supported key sizes and types. The field is not configurable.
KeyType	A string indicating the key type of the template as returned by the CA. See details under *KeySize*. See the *KeyAlgorithms* field for the complete list of supported key sizes and types. The field is not configurable.
ForestRoot	A string indicating the forest root of the template. For Microsoft templates, this field is populated from Active Directory and is not configurable. **Note:** The ForestRoot has been replaced by the ConfigurationTenant from release 10, but is retained for backwards compatibility.
ConfigurationTenant	A string indicating the configuration tenant of the template. For Microsoft templates, this field is populated from Active Directory. For EJBCA templates, this field is populated from the Keyfactor Command CA record. The field is not configurable.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>FriendlyName</td><td>A string indicating the Keyfactor Command friendly name of the template. Template friendly names, if configured, appear in the dropdowns for PFX enrollment, CSR enrollment, and CSR generation in place of the template names. This can be useful in environments where the template names are long or not very human readable.</td></tr>
<tr><td>KeyRetention</td><td>A string indicating the type of key retention certificates enrolled with this template will use to store their private key in Keyfactor Command. The key retention object contains the following parameters:<table><tr><th>Value</th><th>Description</th></tr><tr><td>None</td><td>The private key will not be retained.</td></tr><tr><td>Indefinite</td><td>The private key will be retained until it is explicitly deleted.</td></tr><tr><td>AfterExpiration</td><td>The private key will be retained until the specified number of days after the certificate expires (KeyRetentionDays), at which point it will be scheduled for deletion.</td></tr><tr><td>FromIssuance</td><td>The private key will be retained until the specified number of days after the date on which the certificate was issued (KeyRetentionDays), at which point it will be scheduled for deletion.</td></tr></table></td></tr>
<tr><td>KeyRetentionDays</td><td>An integer indicating the number of days a certificate's private key will be retained in Keyfactor Command before being scheduled for deletion, if private key retention is enabled.</td></tr>
<tr><td>KeyArchival</td><td>A Boolean indicating whether the template has been configured with the key archival setting in Active Directory (true) or not (false). This is a reference field and is not configurable.</td></tr>
<tr><td>EnrollmentFields</td><td>An object containing custom enrollment fields. These are configured on a per-template basis to allow you to submit custom fields with CSR enrollments and PFX enrollments to supply custom request attributes to the CA during the enrollment process. This functionality offers such benefits as:
Preventing users from requesting invalid certificates, based on your specific certificate requirements per template.
Providing additional information to the CA with the CSR.
Once created on the template, these values are shown in Keyfactor Command on the PFX and CSR enrollment pages in the Additional Enrollment Fields section. The fields are mandatory during enrollment. The data will appear on the CA / Issued</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
Certificates attribute tab for certificates enrolled with a template configured with Keyfactor Command enrollment fields.

Note: These are not metadata fields, so they are not stored in the Keyfactor Command database, but simply passed through to the CA. The CA in turn could, via a gateway or policy module, use this data to perform required actions.

The enrollment fields object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the custom enrollment field.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the custom enrollment field. This name will appear on the enrollment pages.</td></tr>
<tr><td>Options</td><td>For multiple choice values, an array of strings containing the value choices.</td></tr>
<tr><td>DataType</td><td>An integer indicating the parameter type. The options are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>String: A free-form data entry field.</td></tr>
<tr><td>2</td><td>Multiple Choice: Provides a list of acceptable values for the field. The multiple choice values are provided in the Options parameter.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>MetadataFields</td><td>
An array of objects containing template-level metadata field settings. Template-level metadata field configurations can override global metadata field configurations in these possible ways:

Configuration on the metadata field of required, optional or hidden.
The default value for the metadata field.
A regular expression defined for the field (string fields only) against which entered data will be validated along with its associated message.
For fields of data type multiple choice, the list of values that appear in multiple choice dropdowns.

Metadata field settings defined on a template apply to enrollments made with that template only. Template-level metadata field settings, if defined, take precedence over global-level metadata field settings.
</td></tr>
</table>

Name	Description
	The metadata fields object contains the following parameters:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the template-specific metadata setting.
DefaultValue	A string containing the default value defined for the metadata field for the specific template.
MetadataId	An integer indicating the global metadata field associated with the template-specific settings.
Validation	A string containing the template-specific regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the *Message* field. For example: `^[a-zA-Z0-9'_\.\-]*@(keyexample\.org|keyexample\.com)$` This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either "@keyexample.org" or "keyexample.com". This field is only supported for metadata fields with data type *string*.
Enrollment	An integer that indicates how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:

Value	Description
0	Optional Users have the option to either enter a value or not enter a value in the field.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr>
<tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr>
</table>
</td></tr>
<tr><td>Message</td><td>A string containing a message to present when a user enters information in a metadata field that does not match the template-specific regular expression (Validation field).</td></tr>
</table>
</td></tr>
<tr><td>AllowedEnrollmentTypes</td><td>An integer indicating the type of enrollment allowed for the certificate template. Setting these options causes the template to appear in dropdowns in the corresponding section of the Management Portal. In the case of CSR Enrollment and PFX Enrollment, the templates only appear in dropdowns on the enrollment pages if they are available for enrollment from a CA also configured for enrollment within Keyfactor Command. See Adding or Modifying a CA Record on page 354 for more information. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>CSR Enrollment & PFX Enrollment</td></tr>
<tr><td>4</td><td>CSR Generation</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>5</td><td>CSR Generation & PFX Enrollment</td></tr>
<tr><td>6</td><td>CSR Generation & CSR Enrollment</td></tr>
<tr><td>7</td><td>CSR Enrollment, PFX Enrollment & CSR Generation</td></tr>
</table>
</td></tr>
<tr><td>TemplateRegexes</td><td>An array of objects containing individual template-level regular expressions against which to validate the subject data. Regular expressions defined on a template apply to enrollments made with that template only. Template-level regular expressions, if defined, take precedence over system-wide regular expressions. For more information about system-wide regular expressions, see <a>GET Templates Settings on page 2395. The template regular expression object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>In integer indicating the Keyfactor Command reference ID of the certificate template the regular expression is associated with.</td></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see <a>GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name

Description

Subject Part	Example

	portion of the field made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least one character in the Common Name field in the enrollment pages.

O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`

Name	Description
	Name / **Description**

Name

Description

Subject Part	Example
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lower-

Name	Description
	Name / **Description**

Subject Part	Example
	case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`

IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers:
	`^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$`
	This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:
	`^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`

IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>MAIL (Subject Alternative Name: Email)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:
<pre>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</pre></td></tr>
<tr><td>UPN (Subject Alternative Name: User Principal Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:
<pre>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</pre></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.

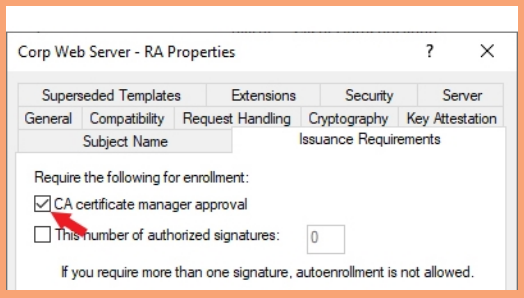
Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
</td></tr>
<tr><td>TemplateDefaults</td><td>An array of objects containing individual template-level template default settings. Template defaults defined on a template apply to enrollments made with that template only. Template-level defaults, if defined, take precedence over system-wide template defaults. For more information about system-wide template defaults,</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>see GET Templates Settings on page 2395. The template default object contains the following parameters:<table><tr><th>Value</th><th>Description</th></tr><tr><td>SubjectPart</td><td>A string indicating the portion of the subject the default applies to (e.g. L for City/Locality).
Use the GET /Templates/SubjectParts method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.</td></tr><tr><td>Value</td><td>A string containing the value to assign as the default for that subject part (e.g. Chicago).</td></tr></table></td></tr>
<tr><td>TemplatePolicy</td><td>An object containing the individual template-level template policy settings. Template policies defined on a template apply to enrollments made with that template only. Template-level policies, if defined, take precedence over system-wide template policies. For more information about system-wide template policies, see GET Templates Settings on page 2395. The template policy object contains the following parameters:<table><tr><th>Value</th><th>Description</th></tr><tr><td>TemplateId</td><td>The Keyfactor Command reference ID of the certificate template the policy is associated with.</td></tr><tr><td>AllowKeyReuse</td><td>A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option applies to certificate renewals. By default, this is set to true at a system-wide level.</td></tr><tr><td>AllowWildcards</td><td>A Boolean that indicates whether wildcards are allowed (true) or not (false). By default, this is set to true at a system-wide level.</td></tr><tr><td>RFCEnforcement</td><td>A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to true, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replic-</td></tr></table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>ated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to *false* at a system-wide level.</td></tr><tr><td>KeyInfo</td><td>An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate. Key info includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>ECDSA</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. - curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command. </td></tr><tr><td>RSA</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. - curves: There are no curves for this type of key. </td></tr><tr><td>Ed448</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor </td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Command.
• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed25519</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>KeyAlgorithms</td><td>An object containing the key algorithms defined for the template as reported by the CA. This information indicates all the algorithms that could possibly be supported when the template is used for enrollment. Template policy within Keyfactor Command might limit this. The key algorithm parameters are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>An integer indicating the ID of the template in Keyfactor Command.</td></tr>
<tr><td>KeyInfo</td><td>An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ECDSA</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command.</td></tr>
<tr><td>RSA</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed448</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed25519</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>UseAllowedRequesters</td><td>A Boolean that indicates whether the Restrict Allowed Requesters option should be enabled (true) or not (false). The Restrict Allowed Requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command using this template. This is typically used for EJBCA templates and Microsoft templates that are not in the local Active Directory forest, since in these cases, Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates; this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the template level</td></tr>
</table>

Name	Description
	on a Microsoft CA. In addition to granting permissions at the template level, you need enable the Restrict Allowed Requesters option to grant permissions at the CA level. See Adding or Modifying a CA Record on page 354 for more information.
AllowedRequesters	An array of strings containing the list of Keyfactor Command security roles—as strings—that have been granted enroll permission on the template.
DisplayName	A string indicating the Keyfactor Command display name of the template. If a template friendly name is configured, this is used as the display name. If not, the template name is used. The display name appears in the dropdowns for PFX enrollment, CSR enrollment, and CSR generation. The display name is a generated field and is not directly configurable.
RFCEnforcement	A Boolean indicating whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to *true*, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to *false* at a system-wide level.
RequiresApproval	A Boolean indicating whether the template has been configured with the Microsoft *CA certificate manager approval* option enabled (*true*) or not (*false*). **Important:** Any templates that are configured on the Microsoft CA Issuance Requirements tab for *CA certificate manager approval* cannot be used for enrollment and associated alerting in Keyfactor Command without configuring private key retention. Any of the enabled private key retention settings (settings other than none as described for *KeyRetention*) will allow a template requiring manager approval to work with Keyfactor Command PFX and CSR enrollment.  *Figure 437: Microsoft Issuance Requirements on a Template for Manager Approval*
KeyUsage	An integer indicating the total key usage of the certificate. Key usage is stored in Active Directory as a single value made of a combination of values. The values that

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>make up the key usage value include:
<table><tr><th>Value</th><th>Function</th><th>Description</th></tr><tr><td>0</td><td>None</td><td>No key usage parameters.</td></tr><tr><td>1</td><td>Encipherment Only</td><td>The key can be used for encryption only.</td></tr><tr><td>2</td><td>CRL Signing</td><td>The key can be used to sign a certificate revocation list (CRL).</td></tr><tr><td>4</td><td>Key Certificate Signing</td><td>The key can be used to sign certificates.</td></tr><tr><td>8</td><td>Key Agreement</td><td>The key can be used to determine key agreement, such as a key created using the Diffie-Hellman key agreement algorithm.</td></tr><tr><td>16</td><td>Data Encipher-ment</td><td>The key can be used for data encryption.</td></tr><tr><td>32</td><td>Key Encipherment</td><td>The key can be used for key encryption.</td></tr><tr><td>64</td><td>Nonrepudiation</td><td>The key can be used for authentication.</td></tr><tr><td>128</td><td>Digital Signature</td><td>The key can be used as a digital signa-ture.</td></tr><tr><td>32768</td><td>Decipherment Only</td><td>The key can be used for decryption only.</td></tr></table>
For example, a value of 160 would represent a key usage of digital signature with key encipherment. A value of 224 would add nonrepudiation to those.</td></tr>
<tr><td>Exten-dedKeyUsages</td><td>An array of objects containing the extended key usage information for the template. This field is populated from the CA and is not configurable. The extended key usage object contains the following parameters:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the ID of the extended key usage in Active Directory.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Oid</td><td>A string containing the object ID of the extended key usage.</td></tr>
<tr><td>DisplayName</td><td>A string specifying the display name of the extended key usage (e.g. Server Authentication).</td></tr>
</table>
</td></tr>
<tr><td>Curve</td><td>A string indicating the friendly name of the elliptic curve algorithm configured for the template returned from the CA, for ECC templates. Possible values include:
P-256
1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1
P-384
1.3.132.0.34 = P-384/secp384r1
P-521
1.3.132.0.35 = P-521/secp521r1
If the template supports more than one curve, this field contains the minimum curve value.</td></tr>
<tr><td>AllowOneClick-Renewals</td><td>A Boolean indicating whether One-Click Renewal will be allowed for certificate renewals requested with this template (true) or not (false).
If you wish to use One-Click Renewal for certificates, the Allow One-Click Renewals option must be enabled in both the templates and CAs to which you want One-Click Renewal to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354). For more information about one-click renewals, see Renew on page 69.</td></tr>
<tr><td>KeyTypes</td><td>A string containing a comma-delimited list of the key sizes and types supported for the template returned from the CA as they are displayed in the Management Portal templates grid. Possible values include RSA 2048, ECC P-384, Ed25519, and Ed448.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.38.2 GET Templates Settings

The GET /Templates/Settings method is used to retrieve the global template policy settings Keyfactor Command. This method returns HTTP 200 OK on a success with details about the global template policy settings. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: Template policies may also be set at an individual template level to apply to a single template (see PUT Templates on page 2435). Template policies set at the individual template level take precedence over template policies set at the global level.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_templates/read/

Table 771: GET Templates Settings Response Data

Name	Description
TemplateRegexes	An array of objects containing the system-wide template regular expression settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Regular expression details are:

Name	Description
SubjectPart	A string indicating the portion of the subject the regular expression applies to (e.g. CN).
RegEx	A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated. Use the *GET /Templates/SubjectParts* method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts. The following are some regular expression examples:

Subject Part	Example

CN (Common Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least one char-

Name	Description
	Name / **Description**

Subject Part	Example
	acter in the Common Name field in the enrollment pages.
O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the

Name	Description
	Name / **Description**

Subject Part	Example
	country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:`

Name	Description
	Name / **Description**

Name	Description
	Subject Part / **Example**

Subject Part	Example

	`[0-9]{1,3})$`
	This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:
	`^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`

IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
MAIL (Subject Alternative Name: Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
UPN (Subject Alternative Name: User Principal Name)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td></td><td><pre>^[a-zA-Z0-9'_
\.\-]*@keyexample\.com$</pre></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.

Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
</td></tr>
<tr><td>TemplateDefaults</td><td>An array of objects containing the system-wide template default settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Template default details are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the default applies to (e.g. L for City/Locality).</td></tr>
<tr><td>Value</td><td>A string containing the value to assign as the default for that subject part (e.g. Chicago).</td></tr>
</table>

Note: See also the Subject Format application setting, which takes precedence over enrollment defaults at both the system-wide and template level (see Application Settings: Enrollment Tab on page 609 in the Keyfactor Command Reference Guide) but does not apply to enrollment requests done through the Keyfactor API.</td></tr>
<tr><td>TemplatePolicy</td><td>An object containing the system-wide template policy settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Template policy details are:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>AllowKeyReuse</td><td>A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option allows to certificate renewals.</td></tr><tr><td>AllowWildcards</td><td>A Boolean that indicates whether wildcards are allowed (true) or not (false).</td></tr><tr><td>RFCEnforcement</td><td>A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to true, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set.</td></tr><tr><td>KeyInfo</td><td>An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate. Key info includes:<table><tr><th>Name</th><th>Description</th></tr><tr><td>ECDSA</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. - curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command. </td></tr><tr><td>RSA</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. </td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed448</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed25519</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.38.3 PUT Templates Settings

The PUT /Templates/Settings method is used to create or update the global template policy settings in Keyfactor Command. This method returns HTTP 200 OK on a success with details about the template policy settings.

Tip: Template policies may also be set at an individual template level to apply to a single template (see PUT Templates on page 2435). Template policies set at the individual template level take precedence over template policies set at the global level.

Note: Global template settings replaced and expanded upon select enrollment-related applications settings in release 10.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_templates/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 772: PUT Templates Settings Input Parameters

Name	Description
TemplateRegexes	An array of objects containing the system-wide template regular expression settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Regular expression details are:

Name	Description
SubjectPart	A string indicating the portion of the subject the regular expression applies to (e.g. CN).
RegEx	A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated. Use the *GET /Templates/SubjectParts* method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts. The following are some regular expression examples:

Subject Part	Example

CN (Common Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least one char-

Name	Description
	Name / **Description**

Subject Part	Example
	acter in the Common Name field in the enrollment pages.
O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the

Name	Description
	Name / **Description**

Subject Part	Example
	country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:`

Name	Description
	Name / **Description**

Name	Description
	Subject Part / **Example**

Subject Part	Example

	`[0-9]{1,3})$`
	This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:
	`^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`

IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
MAIL (Subject Alternative Name: Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
UPN (Subject Alternative Name: User Principal Name)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td></td><td><pre>^[a-zA-Z0-9'_
\.\-]*@keyexample\.com$</pre></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.

Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
For example:
<pre>"TemplateRegexes": [
   {
      "SubjectPart": "O",
      "Regex": "^(?:Key Example Company|Key Example\, Inc\.)$",
      "Error": "Organization must be Key Example, Inc or Key Example Company."
   }
]</pre>
</td></tr>
<tr><td>TemplateDefaults</td><td>An array of objects containing the system-wide template default settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Template default details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the default applies to (e.g. L for City/Locality).</td></tr>
<tr><td>Value</td><td>A string containing the value to assign as the default for that subject part (e.g. Chicago).</td></tr>
</table>
For example:</td></tr>
</table>

Name	Description

	<pre>"TemplateDefaults": [
	  {
	  "SubjectPart": "L",
	  "Value": "Denver"
	  },
	  {
	  "SubjectPart": "ST",
	  "Value": "Colorado"
	  }
	]</pre>
	Note: See also the *Subject Format* application setting, which takes precedence over enrollment defaults at both the system-wide and template level (see Application Settings: Enrollment Tab on page 609 in the *Keyfactor Command Reference Guide*) but does not apply to enrollment requests done through the Keyfactor API.

TemplatePolicy	An object containing the system-wide template policy settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Template policy details are:

Name	Description
AllowKeyReuse	A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option allows to certificate renewals.
AllowWildcards	A Boolean that indicates whether wildcards are allowed (true) or not (false).
RFCEnforcement	A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to *true*, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set.
KeyInfo	An object containing the supported key types for the

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>template along with the bit lengths and/or curves for the key types as appropriate. Key info includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ECDSA</td><td>An object containing two arrays:
bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command.
ECC curves may be specified using the well-known OIDs for ECC algorithms or by friendly name. Well-known OIDs include:
1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1
1.3.132.0.34 = P-384/secp384r1
1.3.132.0.35 = P-521/secp521r1
When specifying by friendly name, do not include a slash (use “P-256”, not “P-256/prime256v1/secp256r1”).</td></tr>
<tr><td>RSA</td><td>An object containing two arrays:
bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
curves: There are no curves for this type of key.
Keyfactor Command supports key sizes 2048, 3072, 4096, 6144, 8192, and 16384.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Ed448</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. - curves: There are no curves for this type of key. </td></tr><tr><td>Ed25519</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. - curves: There are no curves for this type of key. </td></tr></table></td></tr></table>

For example:

```
"TemplatePolicy": {
   "AllowKeyReuse": false,
   "AllowWildcards": true,
   "RFCEnforcement": true,
   "KeyInfo": {
      "ECDSA": {
         "bit_lengths": [
            256,
            384,
            521
         ],
         "curves": [
            "1.2.840.10045.3.1.7",
            "1.3.132.0.34",
            "1.3.132.0.35"
         ]
      },
      "RSA": {
         "bit_lengths": [
            2048,
            4096
         ],
```

Name	Description

	"curves": []
	},
	"Ed448": {
	"bit_lengths": [
	448
	],
	"curves": []
	},
	"Ed25519": {
	"bit_lengths": [
	255
	],
	"curves": []
	}
	}
	}

Table 773: PUT Templates Settings Response Data

Name	Description
TemplateRegexes	An array of objects containing the system-wide template regular expression settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Regular expression details are:

Name	Description
SubjectPart	A string indicating the portion of the subject the regular expression applies to (e.g. CN).
RegEx	A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated. Use the *GET /Templates/SubjectParts* method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts. The following are some regular expression examples:

Subject Part	Example

CN (Common Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least one char-

Name	Description
	Name / **Description**

Subject Part	Example
	acter in the Common Name field in the enrollment pages.
O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the

Name	Description
	Name / **Description**

Subject Part	Example
	country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:`

Name	Description
	Name / **Description**

Name	Description
	Subject Part / **Example**

Subject Part	Example

	`[0-9]{1,3})$`
	This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:
	`^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`

IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
MAIL (Subject Alternative Name: Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
UPN (Subject Alternative Name: User Principal Name)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td></td><td><pre>^[a-zA-Z0-9'_
\.\-]*@keyexample\.com$</pre></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.

Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
For example:
<pre>"TemplateRegexes": [
   {
      "SubjectPart": "O",
      "Regex": "^(?:Key Example Company|Key Example\, Inc\.)$",
      "Error": "Organization must be Key Example, Inc or Key Example Company."
   }
]</pre>
</td></tr>
<tr><td>TemplateDefaults</td><td>An array of objects containing the system-wide template default settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Template default details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the default applies to (e.g. L for City/Locality).</td></tr>
<tr><td>Value</td><td>A string containing the value to assign as the default for that subject part (e.g. Chicago).</td></tr>
</table>
For example:</td></tr>
</table>

Name	Description

	<pre>"TemplateDefaults": [
	  {
	  "SubjectPart": "L",
	  "Value": "Denver"
	  },
	  {
	  "SubjectPart": "ST",
	  "Value": "Colorado"
	  }
	]</pre>
	Note: See also the *Subject Format* application setting, which takes precedence over enrollment defaults at both the system-wide and template level (see Application Settings: Enrollment Tab on page 609 in the *Keyfactor Command Reference Guide*) but does not apply to enrollment requests done through the Keyfactor API.

TemplatePolicy	An object containing the system-wide template policy settings. These apply to all enrollments that are not otherwise overridden by individual template settings, including those that do not use a template (e.g. from a standalone CA). Template policy details are:

Name	Description
AllowKeyReuse	A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option allows to certificate renewals.
AllowWildcards	A Boolean that indicates whether wildcards are allowed (true) or not (false).
RFCEnforcement	A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to *true*, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set.
KeyInfo	An object containing the supported key types for the

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>template along with the bit lengths and/or curves for the key types as appropriate. Key info includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ECDSA</td><td>An object containing two arrays:
bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command.
ECC curves may be specified using the well-known OIDs for ECC algorithms or by friendly name. Well-known OIDs include:
1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1
1.3.132.0.34 = P-384/secp384r1
1.3.132.0.35 = P-521/secp521r1
When specifying by friendly name, do not include a slash (use “P-256”, not “P-256/prime256v1/secp256r1”).</td></tr>
<tr><td>RSA</td><td>An object containing two arrays:
bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
curves: There are no curves for this type of key.
Keyfactor Command supports key sizes 2048, 3072, 4096, 6144, 8192, and 16384.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>Ed448</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. - curves: There are no curves for this type of key. </td></tr><tr><td>Ed25519</td><td>An object containing two arrays: - bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. - curves: There are no curves for this type of key. </td></tr></table></td></tr></table>

For example:

```
"TemplatePolicy": {
   "AllowKeyReuse": false,
   "AllowWildcards": true,
   "RFCEnforcement": true,
   "KeyInfo": {
      "ECDSA": {
         "bit_lengths": [
            256,
            384,
            521
         ],
         "curves": [
            "1.2.840.10045.3.1.7",
            "1.3.132.0.34",
            "1.3.132.0.35"
         ]
      },
      "RSA": {
         "bit_lengths": [
            2048,
            4096
         ],
```

Name	Description

	"curves": []
	},
	"Ed448": {
	"bit_lengths": [
	448
	],
	"curves": []
	},
	"Ed25519": {
	"bit_lengths": [
	255
	],
	"curves": []
	}
	}
	}

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.38.4 GET Templates Subject Parts

The GET /Templates/SubjectParts method is used to retrieve a list of the certificate subject parts that are supported for regular expressions (TemplateRegexes) and defaults (TemplateDefaults). This method returns HTTP 200 OK on a success with the list of supported certificate subject part fields. This method has no input parameters other than the standard headers (see Endpoint Common Features on page 849).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_templates/read/

Table 774: GET Templates Subject Parts Response Data

Name	Description
SubjectPart	A string indicating the supported subject part code (e.g. L for City/Locality).
SubjectPartName	A string containing a friendly name for the subject part (e.g. City/Locality).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.38.5 GET Templates

The GET /Templates method is used to retrieve one or more templates from Keyfactor Command. Results can be limited to selected templates using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with details about the specified templates.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_templates/read/

Table 775: GET Templates Input Parameters

Name	In	Description

QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Template Search Feature on page 406. The query fields supported for this endpoint are:
		• AllowedEnrollmentType (1-PFX Enrollment, 2-CSR Enrollment, 3-CSR Generation, 0-None)
		• ConfigurationTenant
		• DisplayName
		• ForestRoot (deprecated)
		• FriendlyName
		• HasPrivateKeyRetention (True, False)
		• IsDefaultTemplate (True, False)
		• ShortName
		Tip: To filter out all the built-in Active Directory templates and display only your custom templates, use the following query:
		`IsDefaultTemplate -eq "false"`
		To filter out all templates that are not configured for either PFX Enrollment or CSR Enrollment, use the following query:
		`AllowedEnrollmentType -eq "3"`
		A value of 1 will filter out all templates except those configured for PFX Enrollment. A value of 2 will filter out all templates except those configured for CSR Enrollment.

PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *CommonName*.

Name	In	Description
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 776: GET Templates Response Data

Name	Description
Id	An integer indicating the ID of the template in Keyfactor Command.
CommonName	A string containing the common name (short name) of the template. This name typically does not contain spaces. For a template created using a Microsoft management tool, this will be the Microsoft template name. For a template generated for an EJBCA CA, this will be built using a naming scheme of *<end entity profile name>_ <certificate profile name>*. This field is populated based on information retrieved from the CA and is not configurable.
TemplateName	A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name. For a template generated for an EJBCA CA, this will be built using a naming scheme of *<end entity profile name> (<certificate profile name>)*. This field is populated based on information retrieved from the CA and is not configurable.
Oid	A string containing the object ID of the template in Active Directory. For Microsoft templates, this field is populated from Active Directory. For EJBCA templates, this field is generated within Keyfactor Command as an object identifier, but does not follow official OID conventions. The field is not configurable.
KeySize	A string indicating the minimum supported key size of the template as returned by the CA. This value is calculated based on the algorithms provided in the template from the CA (see *KeyAlgorithms*). The algorithm key types and sizes are evaluated in order (RSA, ECC, Ed448, and Ed25519) and from these, the minimum type and size is determined. For example, if the template supports RSA, Ed448, and Ed25519, the minimum key type will be evaluated to RSA. Then for that algorithm, the minimum key size returned by the CA will be selected (e.g. 2048 if 2048 and 4096 are returned for RSA). See the *KeyAlgorithms* field for the complete list of supported key sizes and types. The field is not configurable.
KeyType	A string indicating the key type of the template as returned by the CA. See details under *KeySize*. See the *KeyAlgorithms* field for the complete list of supported key sizes and types. The field is not configurable.
ForestRoot	A string indicating the forest root of the template. For Microsoft templates, this field is populated from Active Directory and is not configurable. **Note:** The ForestRoot has been replaced by the ConfigurationTenant from release 10, but is retained for backwards compatibility.
ConfigurationTenant	A string indicating the configuration tenant of the template. For Microsoft templates, this field is populated from Active Directory. For EJBCA templates, this field is populated from the Keyfactor Command CA record. The field is not configurable.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>FriendlyName</td><td>A string indicating the Keyfactor Command friendly name of the template. Template friendly names, if configured, appear in the dropdowns for PFX enrollment, CSR enrollment, and CSR generation in place of the template names. This can be useful in environments where the template names are long or not very human readable.</td></tr>
<tr><td>KeyRetention</td><td>A string indicating the type of key retention certificates enrolled with this template will use to store their private key in Keyfactor Command. The key retention object contains the following parameters:<table><tr><th>Value</th><th>Description</th></tr><tr><td>None</td><td>The private key will not be retained.</td></tr><tr><td>Indefinite</td><td>The private key will be retained until it is explicitly deleted.</td></tr><tr><td>AfterExpiration</td><td>The private key will be retained until the specified number of days after the certificate expires (KeyRetentionDays), at which point it will be scheduled for deletion.</td></tr><tr><td>FromIssuance</td><td>The private key will be retained until the specified number of days after the date on which the certificate was issued (KeyRetentionDays), at which point it will be scheduled for deletion.</td></tr></table></td></tr>
<tr><td>KeyRetentionDays</td><td>An integer indicating the number of days a certificate's private key will be retained in Keyfactor Command before being scheduled for deletion, if private key retention is enabled.</td></tr>
<tr><td>KeyArchival</td><td>A Boolean indicating whether the template has been configured with the key archival setting in Active Directory (true) or not (false). This is a reference field and is not configurable.</td></tr>
<tr><td>EnrollmentFields</td><td>An object containing custom enrollment fields. These are configured on a per-template basis to allow you to submit custom fields with CSR enrollments and PFX enrollments to supply custom request attributes to the CA during the enrollment process. This functionality offers such benefits as:
Preventing users from requesting invalid certificates, based on your specific certificate requirements per template.
Providing additional information to the CA with the CSR.
Once created on the template, these values are shown in Keyfactor Command on the PFX and CSR enrollment pages in the Additional Enrollment Fields section. The fields are mandatory during enrollment. The data will appear on the CA / Issued</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
Certificates attribute tab for certificates enrolled with a template configured with Keyfactor Command enrollment fields.

Note: These are not metadata fields, so they are not stored in the Keyfactor Command database, but simply passed through to the CA. The CA in turn could, via a gateway or policy module, use this data to perform required actions.

The enrollment fields object contains the following parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the custom enrollment field.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the custom enrollment field. This name will appear on the enrollment pages.</td></tr>
<tr><td>Options</td><td>For multiple choice values, an array of strings containing the value choices.</td></tr>
<tr><td>DataType</td><td>An integer indicating the parameter type. The options are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>String: A free-form data entry field.</td></tr>
<tr><td>2</td><td>Multiple Choice: Provides a list of acceptable values for the field. The multiple choice values are provided in the Options parameter.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>AllowedEnrollmentTypes</td><td>An integer indicating the type of enrollment allowed for the certificate template. Setting these options causes the template to appear in dropdowns in the corresponding section of the Management Portal. In the case of CSR Enrollment and PFX Enrollment, the templates only appear in dropdowns on the enrollment pages if they are available for enrollment from a CA also configured for enrollment within Keyfactor Command. See Adding or Modifying a CA Record on page 354 for more information. Possible values are:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>CSR Enrollment & PFX Enrollment</td></tr>
<tr><td>4</td><td>CSR Generation</td></tr>
<tr><td>5</td><td>CSR Generation & PFX Enrollment</td></tr>
<tr><td>6</td><td>CSR Generation & CSR Enrollment</td></tr>
<tr><td>7</td><td>CSR Enrollment, PFX Enrollment & CSR Generation</td></tr>
</table>
</td></tr>
<tr><td>TemplateRegexes</td><td>An array of objects containing individual template-level regular expressions against which to validate the subject data. Regular expressions defined on a template apply to enrollments made with that template only. Template-level regular expressions, if defined, take precedence over system-wide regular expressions. For more information about system-wide regular expressions, see GET Templates Settings on page 2395. The template regular expression object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>In integer indicating the Keyfactor Command reference ID of the certificate template the regular expression is associated with.</td></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Subject Part	Example

CN (Common Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least one character in the Common Name field in the enrollment pages.

O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments:

Name	Description
	Name / **Description**

Subject Part	Example
	`^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`

Name	Description
	Name / **Description**

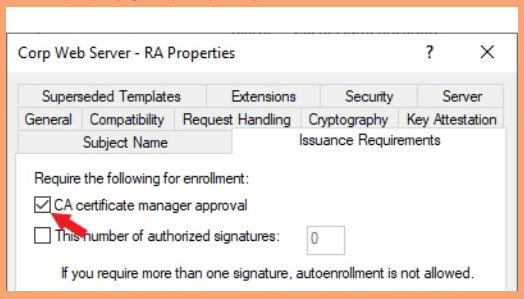
Subject Part	Example
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`

IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers:
	`^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$`
	This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:
	`^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`

IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons:

Name	Description

	Name / **Description**
	
	Subject Part / **Example**
	
	`^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
	
	MAIL (Subject Alternative Name: Email) — This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com":
	`^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
	
	UPN (Subject Alternative Name: User Principal Name) — This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com":
	`^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
	
	Error — A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.
	
	Note: The error message already includes a leading string with the subject part (e.g. "Common Name:" or "Invalid CN provided:" depending on the interface used). Your custom message follows this.

Name	Description
UseAllowedRequesters	A Boolean that indicates whether the Restrict Allowed Requesters option should be enabled (true) or not (false). The Restrict Allowed Requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command using this template. This is typically used for EJBCA templates and Microsoft templates that are not in the local Active Directory forest, since in these cases, Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates; this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the template level on a Microsoft CA. In addition to granting permissions at the template level, you need enable the Restrict Allowed Requesters option to grant permissions at the CA level. See Adding or Modifying a CA Record on page 354 for more information.
AllowedRequesters	An array of strings containing the list of Keyfactor Command security roles—as strings—that have been granted enroll permission on the template.
DisplayName	A string indicating the Keyfactor Command display name of the template. If a template friendly name is configured, this is used as the display name. If not, the template name is used. The display name appears in the dropdowns for PFX enrollment, CSR enrollment, and CSR generation. The display name is a generated field and is not directly configurable.
RequiresApproval	A Boolean indicating whether the template has been configured with the Microsoft *CA certificate manager approval* option enabled (*true*) or not (*false*). **Important:** Any templates that are configured on the Microsoft CA Issuance Requirements tab for *CA certificate manager approval* cannot be used for enrollment and associated alerting in Keyfactor Command without configuring private key retention. Any of the enabled private key retention settings (settings other than none as described for *KeyRetention*) will allow a template requiring manager approval to work with Keyfactor Command PFX and CSR enrollment.  *Figure 438: Microsoft Issuance Requirements on a Template for Manager Approval*
KeyUsage	An integer indicating the total key usage of the certificate. Key usage is stored in Active Directory as a single value made of a combination of values. The values that

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>make up the key usage value include:
<table><tr><th>Value</th><th>Function</th><th>Description</th></tr><tr><td>0</td><td>None</td><td>No key usage parameters.</td></tr><tr><td>1</td><td>Encipherment Only</td><td>The key can be used for encryption only.</td></tr><tr><td>2</td><td>CRL Signing</td><td>The key can be used to sign a certificate revocation list (CRL).</td></tr><tr><td>4</td><td>Key Certificate Signing</td><td>The key can be used to sign certificates.</td></tr><tr><td>8</td><td>Key Agreement</td><td>The key can be used to determine key agreement, such as a key created using the Diffie-Hellman key agreement algorithm.</td></tr><tr><td>16</td><td>Data Encipher-ment</td><td>The key can be used for data encryption.</td></tr><tr><td>32</td><td>Key Encipherment</td><td>The key can be used for key encryption.</td></tr><tr><td>64</td><td>Nonrepudiation</td><td>The key can be used for authentication.</td></tr><tr><td>128</td><td>Digital Signature</td><td>The key can be used as a digital signa-ture.</td></tr><tr><td>32768</td><td>Decipherment Only</td><td>The key can be used for decryption only.</td></tr></table>
For example, a value of 160 would represent a key usage of digital signature with key encipherment. A value of 224 would add nonrepudiation to those.</td></tr>
<tr><td>Exten-dedKeyUsages</td><td>An array of objects containing the extended key usage information for the template. This field is populated from the CA and is not configurable. The extended key usage object contains the following parameters:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the ID of the extended key usage in Active Directory.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Oid</td><td>A string containing the object ID of the extended key usage.</td></tr>
<tr><td>DisplayName</td><td>A string specifying the display name of the extended key usage (e.g. Server Authentication).</td></tr>
</table>
</td></tr>
<tr><td>AllowOneClick-Renewals</td><td>A Boolean indicating whether One-Click Renewal will be allowed for certificate renewals requested with this template (true) or not (false).
If you wish to use One-Click Renewal for certificates, the Allow One-Click Renewals option must be enabled in both the templates and CAs to which you want One-Click Renewal to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354). For more information about one-click renewals, see Renew on page 69.</td></tr>
<tr><td>KeyTypes</td><td>A string containing a comma-delimited list of the key sizes and types supported for the template returned from the CA as they are displayed in the Management Portal templates grid. Possible values include RSA 2048, ECC P-384, Ed25519, and Ed448.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.38.6 PUT Templates

The PUT /Templates method is used to update selected information about a certificate template. This method returns HTTP 200 OK on a success with details about the specified template.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_templates/modify/

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 777: PUT Templates Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td>Body</td><td>Required. An integer indicating the ID of the template in Keyfactor Command.</td></tr>
<tr><td>FriendlyName</td><td>Body</td><td>A string indicating the Keyfactor Command friendly name of the template. Template friendly names, if configured, appear in the dropdowns for PFX enrollment, CSR enrollment, and CSR generation in place of the template names. This can be useful in environments where the template names are long or not very human readable.</td></tr>
<tr><td>KeyRetention</td><td>Body</td><td>A string indicating the type of key retention certificates enrolled with this template will use to store their private key in Keyfactor Command. The key retention object contains the following parameters:<table><tr><th>Value</th><th>Description</th></tr><tr><td>None</td><td>The private key will not be retained.</td></tr><tr><td>Indefinite</td><td>The private key will be retained until it is explicitly deleted.</td></tr><tr><td>AfterExpiration</td><td>The private key will be retained until the specified number of days after the certificate expires (KeyRetentionDays), at which point it will be scheduled for deletion.</td></tr><tr><td>FromIssuance</td><td>The private key will be retained until the specified number of days after the date on which the certificate was issued (KeyRetentionDays), at which point it will be scheduled for deletion.</td></tr></table></td></tr>
<tr><td>KeyRetentionDays</td><td>Body</td><td>An integer indicating the number of days a certificate's private key will be retained in Keyfactor Command before being scheduled for deletion, if private key retention is enabled.</td></tr>
<tr><td>KeyArchival</td><td>Body</td><td>A Boolean indicating whether the template has been configured with the key archival setting in Active Directory (true) or not (false). This is a reference field and is not configurable.</td></tr>
<tr><td>EnrollmentFields</td><td>Body</td><td>An object containing custom enrollment fields. These are configured on a per-template basis to allow you to submit custom fields with CSR enrollments and PFX enrollments to supply custom request attributes to the CA during the enrollment process. This functionality offers such benefits as:
Preventing users from requesting invalid certificates, based on your</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
specific certificate requirements per template.
• Providing additional information to the CA with the CSR.

Once created on the template, these values are shown in Keyfactor Command on the PFX and CSR enrollment pages in the Additional Enrollment Fields section. The fields are mandatory during enrollment. The data will appear on the CA / Issued Certificates attribute tab for certificates enrolled with a template configured with Keyfactor Command enrollment fields.

Note: These are not metadata fields, so they are not stored in the Keyfactor Command database, but simply passed through to the CA. The CA in turn could, via a gateway or policy module, use this data to perform required actions.

The enrollment fields object contains the following parameters:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the custom enrollment field.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the custom enrollment field. This name will appear on the enrollment pages.</td></tr>
<tr><td>Options</td><td>For multiple choice values, an array of strings containing the value choices.</td></tr>
<tr><td>DataType</td><td>An integer indicating the parameter type. The options are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>String: A free-form data entry field.</td></tr>
<tr><td>2</td><td>Multiple Choice: Provides a list of acceptable values for the field. The multiple choice values are provided in the Options parameter.</td></tr>
</table>
</td></tr>
</table>
For example:

<pre>"EnrollmentFields": [
  {</pre>
</td></tr>
</table>

Name	In	Description

		<pre>"Id": 3,
		"Name": "MyCustomField",
		"Options": ["Green","Red","Yellow","Blue"],
		"DataType": 2
		}
		]</pre>

MetadataFields	Body	An array of objects containing template-level metadata field settings. Template-level metadata field configurations can override global metadata field configurations in these possible ways: • Configuration on the metadata field of *required*, *optional* or *hidden*. • The *default value* for the metadata field. • A *regular expression* defined for the field (string fields only) against which entered data will be validated along with its associated message. • For fields of data type *multiple choice*, the list of values that appear in multiple choice dropdowns. Metadata field settings defined on a template apply to enrollments made with that template only. Template-level metadata field settings, if defined, take precedence over global-level metadata field settings. The metadata fields object contains the following parameters:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the template-specific metadata setting.
DefaultValue	A string containing the default value defined for the metadata field for the specific template.
MetadataId	An integer indicating the global metadata field associated with the template-specific settings.
Validation	A string containing the template-specific regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the *Message* field. For example: <pre>^[a-zA-Z0-9'_\.\-]*@</pre>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

<pre>(keyexample\.org|keyexample\.com)$</pre>

This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.
This field is only supported for metadata fields with data type *string*.</td></tr>
<tr><td>Enrollment</td><td>An integer that indicates how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr>
<tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr>
<tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr>
</table></td></tr>
<tr><td>Message</td><td>A string containing a message to present when a</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>user enters information in a metadata field that does not match the template-specific regular expression (Validation field).</td></tr>
</table>
For example:
<pre>"MetadataFields": [
   {
      "Id": 4,
      "DefaultValue": "reggie.wallace@keyexample.com",
      "MetadataId": 4,
      "Validation": "^[a-zA-Z0-9'_\\.\\-]*@(keyexample\\.or-
g|keyexample\\.com)$",
      "Enrollment": 1,
      "Message": "Your email address must be of the form user-
@keyexample.com or fname.lname@keyexample.com."
   },
   {
      "Id": 13,
      "DefaultValue": "E-Business",
      "MetadataId": 5,
      "Validation": "",
      "Enrollment": 0,
      "Message": "",
      "Options": "Accounting,E-Busi-
ness,Executive,HR,IT,Marketing,R&D,Sales"
   }
]</pre>
</td></tr>
<tr><td>AllowedEn-rollmentTypes</td><td>Bod-y</td><td>An integer indicating the type of enrollment allowed for the certificate template. Setting these options causes the template to appear in dropdowns in the corresponding section of the Management Portal. In the case of CSR Enrollment and PFX Enrollment, the templates only appear in dropdowns on the enrollment pages if they are available for enrollment from a CA also configured for enrollment within Keyfactor Command. See <a>Adding or Modifying a CA Record on page 354 for more information. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>CSR Enrollment & PFX Enrollment</td></tr>
<tr><td>4</td><td>CSR Generation</td></tr>
<tr><td>5</td><td>CSR Generation & PFX Enrollment</td></tr>
<tr><td>6</td><td>CSR Generation & CSR Enrollment</td></tr>
<tr><td>7</td><td>CSR Enrollment, PFX Enrollment & CSR Generation</td></tr>
</table>
</td></tr>
<tr><td>TemplateRegexes</td><td>Body</td><td>An array of objects containing individual template-level regular expressions against which to validate the subject data. Regular expressions defined on a template apply to enrollments made with that template only. Template-level regular expressions, if defined, take precedence over system-wide regular expressions. For more information about system-wide regular expressions, see <a>GET Templates Settings on page 2395. The template regular expression object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>In integer indicating the Keyfactor Command reference ID of the certificate template the regular expression is associated with.</td></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see <a>GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:</td></tr>
</table>
</td></tr>
</table>

Name	In	Description

		Name / **Description**
		Subject Part / **Example**
		CN (Common Name) — This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
		`^[a-zA-Z0-9'_\.\-]*\.keyexample\.com$`
		The default value for the Common Name regular expression is:
		`.+`
		This requires entry of at least one character in the Common Name field in the enrollment pages.
		O (Organization) — This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”:
		`^(?:Key Example Inc|Key Example|Key Example, Inc\.)$`
		The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.

Name	In	Description
		Name / **Description**

Subject Part	Example
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:

Name	In	Description

		Name / **Description**
		Subject Part / **Example**
		`^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
		DNS (Subject Alternative Name: DNS Name) — This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, upper-case letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com":
		`^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
		IPv4 (Subject Alternative Name: IPv4 Address) — This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers:
		`^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$`
		This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:
		`^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>IPv6 (Subject Alternative Name: IPv6 Address)</td><td>This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons:
<code>^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$</code></td></tr>
<tr><td>MAIL (Subject Alternative Name: Email)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:
<code>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</code></td></tr>
<tr><td>UPN (Subject Alternative Name: User Principal Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:
<code>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</code></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
For example:
<pre>"TemplateRegexes": [
   {
      "TemplateId": 57,
      "SubjectPart": "O",
      "Regex": "^(?:Key Example Company|Key Example\, Inc\.)$",
      "Error": "Organization must be Key Example, Inc or Key Example Company."
   }
]</pre>
</td></tr>
<tr><td>TemplateDefaults</td><td>Body</td><td>
An array of objects containing individual template-level template default settings. Template defaults defined on a template apply to enrollments made with that template only. Template-level defaults, if defined, take precedence over system-wide template defaults. For more information about system-wide template defaults, see <a>GET Templates Settings on page 2395. The template default object contains the following parameters:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the default applies to (e.g. L for City/Locality).
Use the GET /Templates/SubjectParts method (see <a>GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.</td></tr>
<tr><td>Value</td><td>A string containing the value to assign as the default for that subject part (e.g. Chicago).</td></tr>
</table>
For example:
<pre>"TemplateDefaults": [
   {</pre>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>

<pre>
      "SubjectPart": "L",
      "Value": "Denver"
   },
   {
      "SubjectPart": "ST",
      "Value": "Colorado"
   }
]
</pre>

</td></tr>
<tr><td>TemplatePolicy</td><td>Body</td><td>An object containing the individual template-level template policy settings. Template policies defined on a template apply to enrollments made with that template only. Template-level policies, if defined, take precedence over system-wide template policies. For more information about system-wide template policies, see <a>GET Templates Settings on page 2395. The template policy object contains the following parameters:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>The Keyfactor Command reference ID of the certificate template the policy is associated with.</td></tr>
<tr><td>AllowKeyReuse</td><td>A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option applies to certificate renewals. By default, this is set to true at a system-wide level.</td></tr>
<tr><td>AllowWildcards</td><td>A Boolean that indicates whether wildcards are allowed (true) or not (false). By default, this is set to true at a system-wide level.</td></tr>
<tr><td>RFCEnforcement</td><td>A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to true, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to false at a system-wide level.</td></tr>
</table>

</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>KeyInfo</td><td>An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate. Key info includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ECDSA</td><td>An object containing two arrays:
bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command.
ECC curves may be specified using the well-known OIDs for ECC algorithms or by friendly name. Well-known OIDs include:
1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1
1.3.132.0.34 = P-384/secp384r1
1.3.132.0.35 = P-521/secp521r1
When specifying by friendly name, do not include a slash (use “P-256”, not “P-256/prime256v1/secp256r1”).</td></tr>
<tr><td>RSA</td><td>An object containing two</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Value / **Description** **Name** / **Description** arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no curves for this type of key. Keyfactor Command supports key sizes 2048, 3072, 4096, 6144, 8192, and 16384. Ed448 — An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no curves for this type of key. Ed25519 — An object containing two arrays: • bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command. • curves: There are no curves for this type of key. For example: `"TemplatePolicy": {`

Name	In	Description

		<pre>"AllowKeyReuse": false,
		"AllowWildcards": true,
		"RFCEnforcement": true,
		"KeyInfo": {
		"ECDSA": {
		"bit_lengths": [
		256,
		384,
		521
		],
		"curves": [
		"1.2.840.10045.3.1.7",
		"1.3.132.0.34",
		"1.3.132.0.35"
		]
		},
		"RSA": null,
		"Ed448": null,
		"Ed25519": null
		}
		}</pre>

UseAllowedRequesters	Body	A Boolean that indicates whether the Restrict Allowed Requesters option should be enabled (true) or not (false). The Restrict Allowed Requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command using this template. This is typically used for EJBCA templates and Microsoft templates that are not in the local Active Directory forest, since in these cases, Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates; this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the template level on a Microsoft CA. In addition to granting permissions at the template level, you need enable the Restrict Allowed Requesters option to grant permissions at the CA level. See Adding or Modifying a CA Record on page 354 for more information.

AllowedRequesters	Body	An array of strings containing the list of Keyfactor Command security roles—as strings—that have been granted enroll permission on the template.
		For example:
		<pre>"AllowedRequesters": [
		"Administrator",
		"Power Users",
		"Revokers"
		]</pre>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RequiresApproval</td><td>Body</td><td>A Boolean indicating whether the template has been configured with the Microsoft CA certificate manager approval option enabled (true) or not (false).

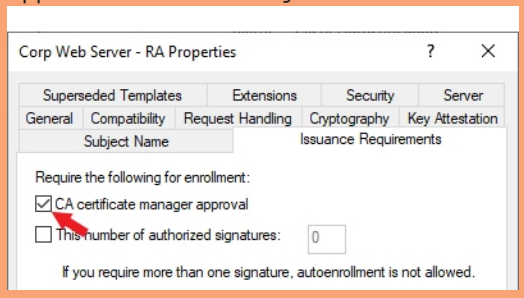
⚠ Important: Any templates that are configured on the Microsoft CA Issuance Requirements tab for CA certificate manager approval cannot be used for enrollment and associated alerting in Keyfactor Command without configuring private key retention. Any of the enabled private key retention settings (settings other than none as described for KeyRetention) will allow a template requiring manager approval to work with Keyfactor Command PFX and CSR enrollment.
 

Figure 439: Microsoft Issuance Requirements on a Template for Manager Approval</td></tr>
<tr><td>KeyUsage</td><td>Body</td><td>An integer indicating the total key usage of the certificate. Key usage is stored in Active Directory as a single value made of a combination of values. The values that make up the key usage value include:

<table>
<tr><th>Value</th><th>Function</th><th>Description</th></tr>
<tr><td>0</td><td>None</td><td>No key usage parameters.</td></tr>
<tr><td>1</td><td>Encipherment Only</td><td>The key can be used for encryption only.</td></tr>
<tr><td>2</td><td>CRL Signing</td><td>The key can be used to sign a certificate revocation list (CRL).</td></tr>
<tr><td>4</td><td>Key Certificate Signing</td><td>The key can be used to sign certificates.</td></tr>
<tr><td>8</td><td>Key Agreement</td><td>The key can be used to determine key agreement, such as a key created using the Diffie-Hellman</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></tr>
<tr><td></td><td></td><td>key agreement algorithm.</td></tr>
<tr><td>16</td><td>Data Encipher-ment</td><td>The key can be used for data encryption.</td></tr>
<tr><td>32</td><td>Key Encipherment</td><td>The key can be used for key encryption.</td></tr>
<tr><td>64</td><td>Nonrepudiation</td><td>The key can be used for authen-tication.</td></tr>
<tr><td>128</td><td>Digital Signature</td><td>The key can be used as a digital signature.</td></tr>
<tr><td>32768</td><td>Decipherment Only</td><td>The key can be used for decryp-tion only.</td></tr>
</table>
For example, a value of 160 would represent a key usage of digital signature with key encipherment. A value of 224 would add nonrepudiation to those.
</td></tr>
<tr><td>AllowOneClick-Renewals</td><td>Bod-y</td><td>A Boolean indicating whether One-Click Renewal will be allowed for certi-ficate renewals requested with this template (true) or not (false).
If you wish to use One-Click Renewal for certificates, the Allow One-Click Renewals option must be enabled in both the templates and CAs to which you want One-Click Renewal to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354). For more information about one-click renewals, see Renew on page 69.</td></tr>
</table>

Table 778: PUT Templates Response Body

Name	Description
Id	An integer indicating the ID of the template in Keyfactor Command.
CommonName	A string containing the common name (short name) of the template. This name typically does not contain spaces. For a template created using a Microsoft management tool, this will be the Microsoft template name. For a template generated for an EJBCA CA, this will be built using a naming scheme of *<end entity profile name>_ <certificate profile name>*. This field is populated based on information retrieved from the CA and is not configurable.
TemplateName	A string containing the name of the template. For a template created using a Microsoft management tool, this will be the Microsoft template display name. For a template generated for an EJBCA CA, this will be built using a naming scheme of *<end entity profile name> (<certificate profile name>)*. This field is populated based on information retrieved from the CA and is not configurable.
Oid	A string containing the object ID of the template in Active Directory. For Microsoft templates, this field is populated from Active Directory. For EJBCA templates, this field is generated within Keyfactor Command as an object identifier, but does not follow official OID conventions. The field is not configurable.
KeySize	A string indicating the minimum supported key size of the template as returned by the CA. This value is calculated based on the algorithms provided in the template from the CA (see *KeyAlgorithms*). The algorithm key types and sizes are evaluated in order (RSA, ECC, Ed448, and Ed25519) and from these, the minimum type and size is determined. For example, if the template supports RSA, Ed448, and Ed25519, the minimum key type will be evaluated to RSA. Then for that algorithm, the minimum key size returned by the CA will be selected (e.g. 2048 if 2048 and 4096 are returned for RSA). See the *KeyAlgorithms* field for the complete list of supported key sizes and types. The field is not configurable.
KeyType	A string indicating the key type of the template as returned by the CA. See details under *KeySize*. See the *KeyAlgorithms* field for the complete list of supported key sizes and types. The field is not configurable.
ForestRoot	A string indicating the forest root of the template. For Microsoft templates, this field is populated from Active Directory and is not configurable. **Note:** The ForestRoot has been replaced by the ConfigurationTenant from release 10, but is retained for backwards compatibility.
ConfigurationTenant	A string indicating the configuration tenant of the template. For Microsoft templates, this field is populated from Active Directory. For EJBCA templates, this field is populated from the Keyfactor Command CA record. The field is not configurable.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>FriendlyName</td><td>A string indicating the Keyfactor Command friendly name of the template. Template friendly names, if configured, appear in the dropdowns for PFX enrollment, CSR enrollment, and CSR generation in place of the template names. This can be useful in environments where the template names are long or not very human readable.</td></tr>
<tr><td>KeyRetention</td><td>A string indicating the type of key retention certificates enrolled with this template will use to store their private key in Keyfactor Command. The key retention object contains the following parameters:<table><tr><th>Value</th><th>Description</th></tr><tr><td>None</td><td>The private key will not be retained.</td></tr><tr><td>Indefinite</td><td>The private key will be retained until it is explicitly deleted.</td></tr><tr><td>AfterExpiration</td><td>The private key will be retained until the specified number of days after the certificate expires (KeyRetentionDays), at which point it will be scheduled for deletion.</td></tr><tr><td>FromIssuance</td><td>The private key will be retained until the specified number of days after the date on which the certificate was issued (KeyRetentionDays), at which point it will be scheduled for deletion.</td></tr></table></td></tr>
<tr><td>KeyRetentionDays</td><td>An integer indicating the number of days a certificate's private key will be retained in Keyfactor Command before being scheduled for deletion, if private key retention is enabled.</td></tr>
<tr><td>KeyArchival</td><td>A Boolean indicating whether the template has been configured with the key archival setting in Active Directory (true) or not (false). This is a reference field and is not configurable.</td></tr>
<tr><td>EnrollmentFields</td><td>An object containing custom enrollment fields. These are configured on a per-template basis to allow you to submit custom fields with CSR enrollments and PFX enrollments to supply custom request attributes to the CA during the enrollment process. This functionality offers such benefits as:
Preventing users from requesting invalid certificates, based on your specific certificate requirements per template.
Providing additional information to the CA with the CSR.
Once created on the template, these values are shown in Keyfactor Command on the PFX and CSR enrollment pages in the Additional Enrollment Fields section. The fields are mandatory during enrollment. The data will appear on the CA / Issued</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
Certificates attribute tab for certificates enrolled with a template configured with Keyfactor Command enrollment fields.

Note: These are not metadata fields, so they are not stored in the Keyfactor Command database, but simply passed through to the CA. The CA in turn could, via a gateway or policy module, use this data to perform required actions.

The enrollment fields object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the custom enrollment field.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the custom enrollment field. This name will appear on the enrollment pages.</td></tr>
<tr><td>Options</td><td>For multiple choice values, an array of strings containing the value choices.</td></tr>
<tr><td>DataType</td><td>An integer indicating the parameter type. The options are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>String: A free-form data entry field.</td></tr>
<tr><td>2</td><td>Multiple Choice: Provides a list of acceptable values for the field. The multiple choice values are provided in the Options parameter.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>MetadataFields</td><td>
An array of objects containing template-level metadata field settings. Template-level metadata field configurations can override global metadata field configurations in these possible ways:

Configuration on the metadata field of required, optional or hidden.
The default value for the metadata field.
A regular expression defined for the field (string fields only) against which entered data will be validated along with its associated message.
For fields of data type multiple choice, the list of values that appear in multiple choice dropdowns.

Metadata field settings defined on a template apply to enrollments made with that template only. Template-level metadata field settings, if defined, take precedence over global-level metadata field settings.
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>The metadata fields object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the template-specific metadata setting.</td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value defined for the metadata field for the specific template.</td></tr>
<tr><td>MetadataId</td><td>An integer indicating the global metadata field associated with the template-specific settings.</td></tr>
<tr><td>Validation</td><td>A string containing the template-specific regular expression against which data entered in a string field will be validated. When a user enters information in a metadata field that does not match the specified regular expression, he or she will see the warning message specified in the Message field. For example:
<pre>^[a-zA-Z0-9'_\.\-]*@
(keyexample\.org|keyexample\.com)$</pre>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either “@keyexample.org” or “keyexample.com”.
This field is only supported for metadata fields with data type string.</td></tr>
<tr><td>Enrollment</td><td>An integer that indicates how metadata fields should be handled on the PFX and CSR Enrollment pages. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Optional
Users have the option to either enter a value or not enter a value in the field.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>Required
Users are required to enter data in the field when populating metadata fields on the PFX and CSR Enrollment pages. The field is not required on the certificate details or Add Certificate page.</td></tr>
<tr><td>2</td><td>Hidden
The field is hidden and does not appear on the PFX and CSR Enrollment pages. This field still appears on the certificate details and the Add Certificate page.</td></tr>
</table>
</td></tr>
<tr><td>Message</td><td>A string containing a message to present when a user enters information in a metadata field that does not match the template-specific regular expression (Validation field).</td></tr>
</table>
</td></tr>
<tr><td>AllowedEnrollmentTypes</td><td>An integer indicating the type of enrollment allowed for the certificate template. Setting these options causes the template to appear in dropdowns in the corresponding section of the Management Portal. In the case of CSR Enrollment and PFX Enrollment, the templates only appear in dropdowns on the enrollment pages if they are available for enrollment from a CA also configured for enrollment within Keyfactor Command. See Adding or Modifying a CA Record on page 354 for more information. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>1</td><td>PFX Enrollment</td></tr>
<tr><td>2</td><td>CSR Enrollment</td></tr>
<tr><td>3</td><td>CSR Enrollment & PFX Enrollment</td></tr>
<tr><td>4</td><td>CSR Generation</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>5</td><td>CSR Generation & PFX Enrollment</td></tr>
<tr><td>6</td><td>CSR Generation & CSR Enrollment</td></tr>
<tr><td>7</td><td>CSR Enrollment, PFX Enrollment & CSR Generation</td></tr>
</table>
</td></tr>
<tr><td>TemplateRegexes</td><td>An array of objects containing individual template-level regular expressions against which to validate the subject data. Regular expressions defined on a template apply to enrollments made with that template only. Template-level regular expressions, if defined, take precedence over system-wide regular expressions. For more information about system-wide regular expressions, see GET Templates Settings on page 2395. The template regular expression object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>In integer indicating the Keyfactor Command reference ID of the certificate template the regular expression is associated with.</td></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name

Description

Subject Part	Example

	portion of the field made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly *.keyexample.com*:
	`^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$`
	The default value for the Common Name regular expression is:
	`.+`
	This requires entry of at least one character in the Common Name field in the enrollment pages.

O (Organization)	This regular expression requires that the organization name entered in the field be one of “Key Example Inc”, “Key Example” or “Key Example Inc.”: `^(?:Key Example Inc|Key Example|Key Example, Inc\.)$` The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression with a slash (”\”) but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`

Name	Description
	Name / **Description**

Name

Description

Subject Part	Example
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”: `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lower-

Name	Description
	Name / **Description**

Subject Part	Example
	case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`

IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers:
	`^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$`
	This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:
	`^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`

IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>MAIL (Subject Alternative Name: Email)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:
<pre>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</pre></td></tr>
<tr><td>UPN (Subject Alternative Name: User Principal Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:
<pre>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</pre></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression.

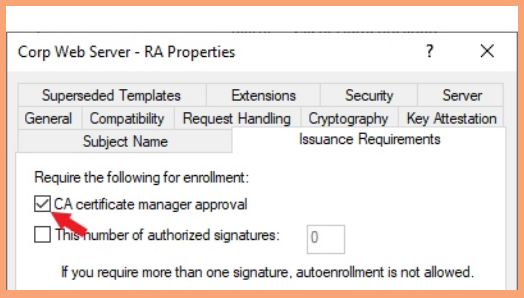
Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
</td></tr>
<tr><td>TemplateDefaults</td><td>An array of objects containing individual template-level template default settings. Template defaults defined on a template apply to enrollments made with that template only. Template-level defaults, if defined, take precedence over system-wide template defaults. For more information about system-wide template defaults,</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>see GET Templates Settings on page 2395. The template default object contains the following parameters:<table><tr><th>Value</th><th>Description</th></tr><tr><td>SubjectPart</td><td>A string indicating the portion of the subject the default applies to (e.g. L for City/Locality).
Use the GET /Templates/SubjectParts method (see GET Templates Subject Parts on page 2421) to retrieve a list of all the supported subject parts.</td></tr><tr><td>Value</td><td>A string containing the value to assign as the default for that subject part (e.g. Chicago).</td></tr></table></td></tr>
<tr><td>TemplatePolicy</td><td>An object containing the individual template-level template policy settings. Template policies defined on a template apply to enrollments made with that template only. Template-level policies, if defined, take precedence over system-wide template policies. For more information about system-wide template policies, see GET Templates Settings on page 2395. The template policy object contains the following parameters:<table><tr><th>Value</th><th>Description</th></tr><tr><td>TemplateId</td><td>The Keyfactor Command reference ID of the certificate template the policy is associated with.</td></tr><tr><td>AllowKeyReuse</td><td>A Boolean that indicates whether private key reuse is allowed (true) or not (false). This option applies to certificate renewals. By default, this is set to true at a system-wide level.</td></tr><tr><td>AllowWildcards</td><td>A Boolean that indicates whether wildcards are allowed (true) or not (false). By default, this is set to true at a system-wide level.</td></tr><tr><td>RFCEnforcement</td><td>A Boolean that indicates whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to true, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replic-</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>ated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to false at a system-wide level.</td></tr>
<tr><td>KeyInfo</td><td>An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate. Key info includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ECDSA</td><td>An object containing two arrays:
bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command.</td></tr>
<tr><td>RSA</td><td>An object containing two arrays:
bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed448</td><td>An object containing two arrays:
bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Command.
• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed25519</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>KeyAlgorithms</td><td>An object containing the key algorithms defined for the template as reported by the CA. This information indicates all the algorithms that could possibly be supported when the template is used for enrollment. Template policy within Keyfactor Command might limit this. The key algorithm parameters are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>An integer indicating the ID of the template in Keyfactor Command.</td></tr>
<tr><td>KeyInfo</td><td>An object containing the supported key types for the template along with the bit lengths and/or curves for the key types as appropriate.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ECDSA</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>• curves: An array of strings indicating the elliptic curve algorithms that are supported for enrollment through Keyfactor Command.</td></tr>
<tr><td>RSA</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed448</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
<tr><td>Ed25519</td><td>An object containing two arrays:
• bit_lengths: An array of integers indicating the key sizes supported for enrollment through Keyfactor Command.
• curves: There are no curves for this type of key.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>UseAllowedRequesters</td><td>A Boolean that indicates whether the Restrict Allowed Requesters option should be enabled (true) or not (false). The Restrict Allowed Requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command using this template. This is typically used for EJBCA templates and Microsoft templates that are not in the local Active Directory forest, since in these cases, Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates; this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the template level</td></tr>
</table>

Name	Description
	on a Microsoft CA. In addition to granting permissions at the template level, you need enable the Restrict Allowed Requesters option to grant permissions at the CA level. See Adding or Modifying a CA Record on page 354 for more information.
AllowedRequesters	An array of strings containing the list of Keyfactor Command security roles—as strings—that have been granted enroll permission on the template.
DisplayName	A string indicating the Keyfactor Command display name of the template. If a template friendly name is configured, this is used as the display name. If not, the template name is used. The display name appears in the dropdowns for PFX enrollment, CSR enrollment, and CSR generation. The display name is a generated field and is not directly configurable.
RFCEnforcement	A Boolean indicating whether RFC 2818 compliance enforcement is enabled (true) or not (false). When this option is set to *true*, certificate enrollments made through Keyfactor Command for this template must include at least one DNS SAN. In the Keyfactor Command Management Portal, this causes the CN entered in PFX enrollment to automatically be replicated as a SAN, which the user can either change or accept. For CSR enrollment, if the CSR does not have a SAN that matches the CN, one will automatically be added to the certificate if this is set. By default, this is set to *false* at a system-wide level.
RequiresApproval	A Boolean indicating whether the template has been configured with the Microsoft *CA certificate manager approval* option enabled (*true*) or not (*false*). **Important:** Any templates that are configured on the Microsoft CA Issuance Requirements tab for *CA certificate manager approval* cannot be used for enrollment and associated alerting in Keyfactor Command without configuring private key retention. Any of the enabled private key retention settings (settings other than none as described for *KeyRetention*) will allow a template requiring manager approval to work with Keyfactor Command PFX and CSR enrollment.  *Figure 440: Microsoft Issuance Requirements on a Template for Manager Approval*
KeyUsage	An integer indicating the total key usage of the certificate. Key usage is stored in Active Directory as a single value made of a combination of values. The values that

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>make up the key usage value include:
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></tr>
<tr><td>0</td><td>None</td><td>No key usage parameters.</td></tr>
<tr><td>1</td><td>Encipherment Only</td><td>The key can be used for encryption only.</td></tr>
<tr><td>2</td><td>CRL Signing</td><td>The key can be used to sign a certificate revocation list (CRL).</td></tr>
<tr><td>4</td><td>Key Certificate Signing</td><td>The key can be used to sign certificates.</td></tr>
<tr><td>8</td><td>Key Agreement</td><td>The key can be used to determine key agreement, such as a key created using the Diffie-Hellman key agreement algorithm.</td></tr>
<tr><td>16</td><td>Data Encipher-ment</td><td>The key can be used for data encryption.</td></tr>
<tr><td>32</td><td>Key Encipherment</td><td>The key can be used for key encryption.</td></tr>
<tr><td>64</td><td>Nonrepudiation</td><td>The key can be used for authentication.</td></tr>
<tr><td>128</td><td>Digital Signature</td><td>The key can be used as a digital signa-ture.</td></tr>
<tr><td>32768</td><td>Decipherment Only</td><td>The key can be used for decryption only.</td></tr>
</table>
For example, a value of 160 would represent a key usage of digital signature with key encipherment. A value of 224 would add nonrepudiation to those.</td></tr>
<tr><td>Exten-dedKeyUsages</td><td>An array of objects containing the extended key usage information for the template. This field is populated from the CA and is not configurable. The extended key usage object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the extended key usage in Active Directory.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Oid</td><td>A string containing the object ID of the extended key usage.</td></tr>
<tr><td>DisplayName</td><td>A string specifying the display name of the extended key usage (e.g. Server Authentication).</td></tr>
</table>
</td></tr>
<tr><td>Curve</td><td>A string indicating the friendly name of the elliptic curve algorithm configured for the template returned from the CA, for ECC templates. Possible values include:
P-256
1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1
P-384
1.3.132.0.34 = P-384/secp384r1
P-521
1.3.132.0.35 = P-521/secp521r1
If the template supports more than one curve, this field contains the minimum curve value.</td></tr>
<tr><td>AllowOneClick-Renewals</td><td>A Boolean indicating whether One-Click Renewal will be allowed for certificate renewals requested with this template (true) or not (false).
If you wish to use One-Click Renewal for certificates, the Allow One-Click Renewals option must be enabled in both the templates and CAs to which you want One-Click Renewal to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354). For more information about one-click renewals, see Renew on page 69.</td></tr>
<tr><td>KeyTypes</td><td>A string containing a comma-delimited list of the key sizes and types supported for the template returned from the CA as they are displayed in the Management Portal templates grid. Possible values include RSA 2048, ECC P-384, Ed25519, and Ed448.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.38.7 POST Templates Import

The POST /Templates/Import method is used to import templates from a specified configuration tenant into Keyfactor Command. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificate_templates/modify/

Table 779: POST Templates Import Input Parameters

Name	Description
ConfigurationTenant	A string indicating the name of the configuration tenant from which to import.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.39 Workflow Certificates

The endpoints in Keyfactor Command that are found under /Workflow/Certificates refer to the process through which certificate requests that are require manager approval at the CA level before issuance are approved or denied. These endpoints provide the ability to obtain a list of pending certificate enrollment requests, and approve or deny current requests. Endpoints are also included to view denied and external validation requests.

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230) are not managed with these endpoints. Instead, refer to the Workflow Definitions and Workflow Instances endpoints (see Workflow Definitions on page 2487 and Workflow Instances on page 2628).

Table 780: Workflow Certificates Endpoints

Endpoint	Method	Description	Link
/Certificates/{id}	GET	Retrieve certificate request information for a single request.	GET Workflow Certificates ID on the next page
/Certificates/Denied	GET	Retrieve a list of denied certificate request(s).	GET Workflow Certi-

Endpoint	Method	Description	Link
			ficates Denied on page 2475
/Certificates/Pending	GET	Retrieve a list of outstanding pending certificate request(s).	GET Workflow Certificates Pending on page 2478
/Certificates/ExternalValidation	GET	Retrieve a list of certificate request(s) requiring external validation.	GET Workflow Certificates External Validation on page 2481
/Certificates/Approve	POST	Approve a list of pending certificate request(s).	POST Workflow Certificates Approve on page 2486
/Certificates/Deny	POST	Deny a list of pending certificate request(s).	POST Workflow Certificates Deny on page 2484

3.6.39.1 GET Workflow Certificates ID

The Workflow GET /Certificates/{id} method is used to return details for a certificate enrollment request stored within Keyfactor Command that requires manager approval at the CA level. This method returns HTTP 200 OK on a success with the specified certificate request. This method will return certificate requests with any state (e.g. Pending, Denied, External Validation).

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*) are not managed with this endpoint. Instead, refer to the Workflow Definitions and Workflow Instances endpoints (see Workflow Definitions on page 2487 and Workflow Instances on page 2628).

Note: Certificate requests that require approval at the CA level are supported only for Microsort CAs and select CA gateways. This feature is not supported for EJBCA CAs. Use workflow for configuring Keyfactor Command-level approvals for EJBCA CAs (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 781: GET Workflow Certificates {id} Input Parameters

Name	In	Description
id	Path	**Required.** An integer indicating the ID of the certificate request to retrieve. Use the *GET /Workflow/Certificates/Pending* method (see GET Workflow Certificates Pending on page 2478) to retrieve a list of all the certificate requests to determine the certificate request ID.

Table 782: GET Workflow Certificates {id} Input Parameters

Name	Description
Id	An integer indicating the reference ID in Keyfactor Command for the certificate request as stored in the Keyfactor Command database. This is not the same as the request ID issued by the CA. This maps to the KeyfactorRequestId parameter for pending certificate request approve and deny actions. **Note:** The reference ID for the certificate request in Keyfactor Command does not necessarily match the reference ID for the issued certificate in Keyfactor Command.
CARequestId	An integer indicating the row index of the certificate request in the certificate authority.
CommonName	A string indicating the common name of the requested certificate.
DistinguishedName	A string indicating the distinguished name of the requested certificate.
SubmissionDate	The date and time at which the certificate request was received, as an ISO-8601 formatted UTC timestamp.
CertificateAuthority	A string indicating the name of the certificate authority from which the certificate was requested in hostname\logical name format. For example: `corpca01.keyexample.com\\CorpIssuingCA1`
Template	A string indicating the name of the template used for the certificate request.
Requester	A string containing the name of the identity that requested the certificate.
State	An integer indicating the request state of the certificate. The possible values are: • Unknown (0) • Active (1) • Revoked (2) • Denied (3) • Failed (4) • Pending (5) • Certificate Authority (6) • Parent Certificate Authority (7) • External Validation (8)
StateString	A string indicating the request state of the certificate (e.g. Pending).

Name	Description
Metadata	An object containing the metadata fields populated for the certificate request.
DenialComment	A string containing the user-provided comment entered when the certificate request was denied.
KeyLength	An integer indicating the key length of the certificate request.
SANs	An array of strings listing the subject alternative name (SAN) elements of the certificate request.
CertStores	An array of objects containing the certificate store locations to which the certificate resulting from the request will be distributed once approved. Certificate store location data includes: **Name** / **Description** EntryName — A string indicating the alias of the certificate in the certificate store. This value will be blank for store types that use information from the issued certificate (e.g. the thumbprint) as the alias until the request is approved and a certificate issued. ClientMachine — A string indicating the machine on which the certificate store is located. StorePath — A string indicating the path on the machine where the certificate store is located. The format of this will vary depending on the type of store.
Curve	A string indicating the OID of the elliptic curve algorithm configured used for the certificate request, for ECC certificate requests. Well-known OIDs include: • 1.2.840.10045.3.1.7 = P-256/prime256v1/secp256r1 • 1.3.132.0.34 = P-384/secp384r1 • 1.3.132.0.35 = P-521/secp521r1
SubjectAltNames	An array of objects indicating the subject alternative name (SAN) elements for the certificate request. SAN data includes: **Name** / **Description** Value — A string indicating the value set for the SAN element. Type — A string indicating the type of SAN element (e.g. DNS Name).

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.39.2 GET Workflow Certificates Denied

The GET /Workflow/Certificates/Denied method is used to return a list of denied certificate enrollment requests stored within Keyfactor Command for requests that required manager approval at the CA level. Results can be limited to selected requests using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with the specified denied certificate requests.

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*) are not managed with this endpoint. Instead, refer to the Workflow Definitions and Workflow Instances endpoints (see Workflow Definitions on page 2487 and Workflow Instances on page 2628).

Note: Certificate requests that require approval at the CA level are supported only for Microsort CAs and select CA gateways. This feature is not supported for EJBCA CAs. Use workflow for configuring Keyfactor Command-level approvals for EJBCA CAs (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 783: GET Workflow Certificates Denied Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • CAHostname • CALogical • CommonName • Requester • RequestType (3-Denied, 5-Pending, 8-External Validation) This method only returns records of type (State) 3. • SubmissionDate • Template **Tip:** For example, for recent denied requests from requester key_service: `SubmissionDate -ge "2023-09-01T00:00:00Z" AND Requester -eq "KEYEXAMPLE\key_service"`
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *CommonName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 784: GET Workflow Certificates Denied Response Data

Name	Description
Id	An integer indicating the reference ID in Keyfactor Command for the certificate request as stored in the Keyfactor Command database. This is not the same as the request ID issued by the CA. This maps to the KeyfactorRequestId parameter for pending certificate request approve and deny actions.
CARequestId	An integer indicating the row index of the certificate request in the certificate authority.
CommonName	A string indicating the common name of the requested certificate.
DistinguishedName	A string indicating the distinguished name of the requested certificate.
SubmissionDate	The date and time at which the certificate request was received, as an ISO-8601 formatted UTC timestamp.
CertificateAuthority	A string indicating the name of the certificate authority from which the certificate was requested in hostname\logical name format. For example: `corpca01.keyexample.com\\CorpIssuingCA1`
Template	A string indicating the name of the template used for the certificate request.
Requester	A string containing the name of the identity that requested the certificate.
State	An integer indicating the request state of the certificate. **Note:** This method returns only requests with state 3 (denied).
StateString	A string indicating the request state of the certificate (e.g. Pending). **Note:** This method returns only requests with a Denied state.
Metadata	An object containing the metadata fields populated for the certificate request.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.39.3 GET Workflow Certificates Pending

The GET /Workflow/Certificates/Pending method is used to return a list of pending certificate enrollment requests stored within Keyfactor Command for requests that require manager approval at the CA level. Results can be limited to selected requests using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with the specified pending certificate requests.

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*) are not managed with this endpoint. Instead, refer to the Workflow Definitions and Workflow Instances endpoints (see Workflow Definitions on page 2487 and Workflow Instances on page 2628).

Note: Certificate requests that require approval at the CA level are supported only for Microsort CAs and select CA gateways. This feature is not supported for EJBCA CAs. Use workflow for configuring Keyfactor Command-level approvals for EJBCA CAs (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 785: GET Workflow Certificates Pending Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • CAHostname • CALogical • CommonName • Requester • RequestType (3-Denied, 5-Pending, 8-External Validation) This method only returns records of type (State) 5. • SubmissionDate • Template **Tip:** For example, for recent pending requests from requester key_service: `SubmissionDate -ge "2023-09-01T00:00:00Z" AND Requester -eq "KEYEXAMPLE\key_service"`
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *CommonName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 786: GET Workflow Certificates Pending Response Data

Name	Description
Id	An integer indicating the reference ID in Keyfactor Command for the certificate request as stored in the Keyfactor Command database. This is not the same as the request ID issued by the CA. This maps to the KeyfactorRequestId parameter for pending certificate request approve and deny actions.
CARequestId	An integer indicating the row index of the certificate request in the certificate authority.
CommonName	A string indicating the common name of the requested certificate.
DistinguishedName	A string indicating the distinguished name of the requested certificate.
SubmissionDate	The date and time at which the certificate request was received, as an ISO-8601 formatted UTC timestamp.
CertificateAuthority	A string indicating the name of the certificate authority from which the certificate was requested in hostname\logical name format. For example: `corpca01.keyexample.com\\CorpIssuingCA1`
Template	A string indicating the name of the template used for the certificate request.
Requester	A string containing the name of the identity that requested the certificate.
State	An integer indicating the request state of the certificate. **Note:** This method returns only requests with state 5 (pending).
StateString	A string indicating the request state of the certificate (e.g. Pending). **Note:** This method returns only requests with a Pending state.
Metadata	An object containing the metadata fields populated for the certificate request.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.39.4 GET Workflow Certificates External Validation

The GET /Workflow/Certificates/ExternalValidation method is used to return a list of certificate enrollment requests requiring external validation (at the public CA level) stored within Keyfactor Command. Results can be limited to selected requests using filtering, and URL parameters can be used to specify paging and the level of information detail. This method returns HTTP 200 OK on a success with the specified certificate requests requiring external validation.

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*) are not managed with this endpoint. Instead, refer to the Workflow Definitions and Workflow Instances endpoints (see Workflow Definitions on page 2487 and Workflow Instances on page 2628).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/monitoring/alerts/read/

Table 787: GET Workflow Certificates External Validation Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Certificate Search Page on page 34. The query fields supported for this endpoint are: • CAHostname • CALogical • CommonName • Requester • RequestType (3-Denied, 5-Pending, 8-External Validation) This method only returns records of type (State) 8. • SubmissionDate • Template **Tip:** For example, for recent external validation requests from requester key_service: `SubmissionDate -ge "2023-09-01T00:00:00Z" AND Requester -eq "KEYEXAMPLE\key_service"`
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *CommonName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 788: GET Workflow Certificates External Validation Response Data

Name	Description
Id	An integer indicating the reference ID in Keyfactor Command for the certificate request as stored in the Keyfactor Command database. This is not the same as the request ID issued by the CA. This maps to the KeyfactorRequestId parameter for pending certificate request approve and deny actions.
CARequestId	An integer indicating the row index of the certificate request in the certificate authority.
CommonName	A string indicating the common name of the requested certificate.
DistinguishedName	A string indicating the distinguished name of the requested certificate.
SubmissionDate	The date and time at which the certificate request was received, as an ISO-8601 formatted UTC timestamp.
CertificateAuthority	A string indicating the name of the certificate authority from which the certificate was requested in hostname\logical name format. For example: `corpca01.keyexample.com\\CorpIssuingCA1`
Template	A string indicating the name of the template used for the certificate request.
Requester	A string containing the name of the identity that requested the certificate.
State	An integer indicating the request state of the certificate. **Note:** This method returns only requests with state 8 (external validation).
StateString	A string indicating the request state of the certificate (e.g. Pending). **Note:** This method returns only requests with an External Validation state.
Metadata	An object containing the metadata fields populated for the certificate request.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.39.5 POST Workflow Certificates Deny

The POST /Workflow/Certificates/Deny method will attempt to deny the provided pending certificate enrollment request(s) that require manager approval at the CA level. This method returns HTTP 200 OK on a success with details about successful, failed and denied denial requests.

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*) are not managed with this endpoint. Instead, refer to the Workflow Definitions and Workflow Instances endpoints (see Workflow Definitions on page 2487 and Workflow Instances on page 2628).

Note: Certificate requests that require approval at the CA level are supported only for Microsort CAs and select CA gateways. This feature is not supported for EJBCA CAs. Use workflow for configuring Keyfactor Command-level approvals for EJBCA CAs (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/requests/manage/

Table 789: POST Workflow Certificates Deny Input Parameters

Name	In	Description
CertificateRequestIds	Body	**Required**. An array of integers indicating the Keyfactor Command certificate request IDs for certificate requests that should be denied in the form: `[23,45,12]` Use the *GET /Workflow/Certificates/Pending* method (see GET Workflow Certificates Pending on page 2478) to retrieve a list of all the pending certificate requests to determine the certificate request's IDs.
Comment	Body	A string providing a comment regarding the denial. This comment can be delivered to the requester or other interested party using a denied request alert.

Table 790: POST Workflow Certificates Deny Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Successes</td><td>An array of strings indicating the successful denial response details. Response details contain the following information:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CAHost</td><td>Host name of the certificate authority to which the certificate enrollment request was submitted.</td></tr>
<tr><td>CALogicalName</td><td>Logical name of the certificate authority to which the certificate enrollment request was submitted.</td></tr>
<tr><td>CARequestId</td><td>The row index of the certificate request in the certificate authority.</td></tr>
<tr><td>KeyfactorRequestId</td><td>An integer indicating the Keyfactor Command reference ID for the requested certificate as stored in the Keyfactor Command database. This is not the same as the request ID issued by the CA. This maps to the Id response parameter for the GET /Workflow/Certificate/Pending method.</td></tr>
<tr><td>Comment</td><td>A comment about the denial. For example, for a deny that succeeds, the comment will be “Successful”. Denies that fail or are denies will have alternate comments (see below).</td></tr>
</table></td></tr>
<tr><td>Failures</td><td>An array of strings indicating the failed approval response details containing the information noted above for successes. Failures of this type are generally exceptions.</td></tr>
<tr><td>Denials</td><td>An array of strings indicating the denial requests that were denied containing the information noted above for successes. Denials are usually the result of insufficient user permissions required to perform the deny.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.39.6 POST Workflow Certificates Approve

The POST /Workflow/Certificates/Approve method will attempt to approve the provided pending certificate enrollment request(s) that require manager approval at the CA level. This method returns HTTP 200 OK on a success with details about successful, failed and denied approval requests.

Note: Certificate requests that require approval at the Keyfactor Command workflow level (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*) are not managed with this endpoint. Instead, refer to the Workflow Definitions and Workflow Instances endpoints (see Workflow Definitions on the next page and Workflow Instances on page 2628).

Note: Certificate requests that require approval at the CA level are supported only for Microsort CAs and select CA gateways. This feature is not supported for EJBCA CAs. Use workflow for configuring Keyfactor Command-level approvals for EJBCA CAs (see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/certificates/requests/manage/

Table 791: POST Workflow Certificates Approve Input Parameters

Name	In	Description
requestIds	Body	**Required**. An array of integers indicating the Keyfactor Command certificate request IDs for certificate requests that should be approved in the form (without parameter name): `[23,45,12]` Use the *GET /Workflow/Certificates/Pending* method (see GET Workflow Certificates Pending on page 2478) to retrieve a list of all the certificate requests to determine the certificate request's IDs.

Table 792: POST Workflow Certificates Approve Response Data

Name	Description
Successes	An array of strings indicating the successful approval response details. Response details contain the following information: **Name** / **Description** CAHost — Host name of the certificate authority to which the certificate enrollment request was submitted. CALogicalName — Logical name of the certificate authority to which the certificate enrollment request was submitted. CARequestId — The row index of the certificate request in the certificate authority. KeyfactorRequestId — An integer indicating the Keyfactor Command reference ID for the requested certificate as stored in the Keyfactor Command database. This is not the same as the request ID issued by the CA. This maps to the Id response parameter for the *GET /Workflow/Certificate/Pending* method. Comment — A reason or description about why the request denials succeeded, failed or were denied.
Failures	An array of strings indicating the failed approval response details containing the information noted above for successes. Failures of this type are generally exceptions.
Denials	An array of strings indicating the approval requests that were denied containing the information noted above for successes. Denials are usually the result of insufficient user permissions required to perform the approval.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40 Workflow Definitions

The Workflow Definitions component of the Keyfactor API includes methods necessary to programmatically create, edit, retrieve, and test workflow definitions. There are two types of workflow definition:

- Global

  The global workflow definitions are built into the product and cannot be deleted, though they can be modified to add workflow steps, if desired. Global workflow definitions do not have a specific associated *key*—in the case of the currently available workflows, this is a *certificate template*—and apply to all requests of the workflow's type (e.g. enrollment) that are not otherwise handled by a custom workflow specifying a key.

- Custom

  Custom workflow definitions are any additional workflow definitions you define beyond the built-in ones. Custom workflows are associated with a specific *key* (certificate template or certificate collection) and each workflow only applies to requests made using that key.

All enrollment, certificate renewal, and revocation requests go through workflow even if you haven't created any workflow steps or added any custom workflow definitions. In the absence of customization, the global workflow definitions are used. Monitoring certificate collections on a periodic basis for certificates that change membership status based on the query criteria of a specified certificate collection can be configured to flow through workflow as well, but there are no global workflows for these.

For more information about workflows, see Workflow Definitions on page 230 in the *Keyfactor Command Reference Guide*.

Table 793: Workflow Definitions Endpoints

Endpoint	Method	Description	Link
/Steps/{extensionName}	GET	Returns information about the structure of the workflow definition step with the specified name.	GET Workflow Definitions Steps Extension Name on the next page
/{definitionId}	DELETE	Deletes the workflow definition with the specified GUID.	DELETE Workflow Definitions Definition ID on page 2494
/{definitionId}	GET	Returns details of the workflow definition, including steps, for the workflow with the specified GUID.	GET Workflow Definitions Definition ID on page 2494
/{definitionId}	PUT	Updates the name and description of the workflow definition with the specified GUID.	PUT Workflow Definitions Definition ID on page 2518
/	GET	Returns a list of workflow definitions, without steps.	GET Workflow Definitions on page 2542
/	POST	Creates a new workflow definition, without steps.	POST Workflow Definitions on page 2545
/Steps	GET	Returns information about the	GET Workflow Defin-

Endpoint	Method	Description	Link
		structure of the workflow definitions.	itions Steps on page 2570
/Types	GET	Returns a list of the defined workflow definition types.	GET Workflow Definitions Types on page 2576
/{definitionId}/Steps	PUT	Updates the workflow definition with the specified GUID to add new steps or modify existing steps.	PUT Workflow Definitions Definition ID Steps on page 2579
/{definitionId}/Publish	POST	Publishes the workflow definition with the specified GUID to activate it for use.	POST Workflow Definitions Definition ID Publish on page 2604

3.6.40.1 GET Workflow Definitions Steps Extension Name

The GET /Workflow/Definitions/Steps/{extensionName} method is used to retrieve the workflow definition step structure for the step with the specified extensionName. Its primary use case is to populate the UI dialog in which step information is configured. When you are developing a custom workflow step, it can be used to confirm that the workflow step will display correctly in the UI. This method returns HTTP 200 OK on a success with information about the structure of the workflow definition step.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/read/

Table 794: GET Workflow Definitions Steps {extensionName} Input Parameters

Name	In	Description
extensionName	Path	**Required**. A string indicating the *extensionName* of the workflow definition step to retrieve. Use the *GET /Workflow/Definitions/Steps* method (see GET Workflow Definitions Steps on page 2570) to retrieve a list of all the workflow definition steps to determine the extensionName.

Table 795: GET Workflow Definitions Steps {extensionName} Response Data

Name	Description
DisplayName	A string indicating the display name of the workflow definition step.
ExtensionName	A string indicating the extension name of the workflow definition step. The built-in extension names are: • **Email** Send an email message. This is a separate email message from those typically sent as part of a *Require Approval* step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request. • **PowerShell** Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including: • ConvertFrom-Csv • ConvertFrom-Json • ConvertFrom-Markdown • ConvertFrom-SddlString • ConvertFrom-StringData • ConvertTo-Csv • ConvertTo-Html • ConvertTo-Json • ConvertTo-Xml • ForEach-Object • Get-Command • Where-Object • **CustomPowerShell** Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See Extensions Scripts on page 1704 for

Name	Description
	adding scripts to the database. • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step can require approval from just one approver or multiple approvers. The workflow will be suspended at this point until the correct number of approvals from users with the correct security roles is received or until one deny is received before continuing to the next step. As part of this step, an email message is sent indicating whether the step was approved or denied—typically to the requester. This step does not include logic to send an email initiating the approval process (letting users know something needs approval). Use an *Email* type step for this. **Important:** Workflows are not supported with CA delegation when they contain steps that require approval. For more information, see the CA configuration Authorization Methods Tab on page 367. **Note:** The users that you send email to initiating the approval process must be members of a security role that is allowed to submit signals (approve/deny) for the workflow in order to approve or deny the request. **Tip:** The workflow builder does not include a step to send a notification to the requester of a certificate on an enrollment once the certificate is issued by the CA (as opposed to approved in Keyfactor Command). Use the issued alerts for this (see Issued Request Alert Operations on page 189). • **RESTRequest** Run a REST (API) request using Active Directory as an identity provider and Basic or Windows authentication. The REST request contents are embedded within the step. It does not call out to an external file. • **OAuthRESTRequest** Run a REST (API) request using an identity provider other than Active Directory and Token authentication. The REST request contents are embedded within the step. It does not call out to an external file.

Name	Description
	• **EnrollmentAgent (Enrollment Only)** On an enrollment (either CSR or PFX), create a resigned CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) to modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types or a custom step type. The step creates a new CSR using the same public key as the original CSR using the updated SAN and/or subject values. It signs the new CSR with the certificate provided in the step's configuration. For this type of step you will need an enrollment agent certificate available as a PKCS#12 (.PFX) file with included private key to import into Keyfactor Command. This can be a user certificate or a computer certificate (e.g. generated from a copy of the *Enrollment Agent* template or the *Enrollment Agent (Computer)* template) and must have a Certificate Request Agent EKU. Note that the built-in *Enrollment Agent* and *Enrollment Agent (Computer)* templates do not allow private keys to be exported by default. You will need a template that allows private key export or will need to manually override private key export to create a certificate with an exportable private key in order to create a PKCS#12 (.PFX) file. ⚠ **Important:** This step applies to Microsoft CAs only. If this step is added to workflow for requests directed to an EJBCA CA, it will fail on enrollment. Note that EJBCA supports submission of updated SAN or subject details as part of standard functionality. • **SubjectFormatter (Enrollment Only)** On an enrollment done through the Keyfactor Windows Enrollment Gateway using a client-side template configured with the *Build from this Active Directory information* option on the template, this workflow step handles formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template

Name	Description
	configured with the subject as *Build from this Active Directory information* must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. **Important:** The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the *Keyfactor Windows Enrollment Gateway Installation and Configuration Guide*. • **EnrollStep** Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. • **NOOPStep** An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. • **RevokeStep** Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately following the EndNOOP step.
Outputs	An array of strings containing the outputs for the workflow definition step. For the built-in steps, the only output is an indicator for the next step in the workflow.
ConfigurationParametersDefinition	An object containing the configuration parameters for the workflow definition step. These will vary depending on the step.
SignalsDefinition	An object containing the signals defined for the workflow definition step. These will vary depending on the step.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.2 DELETE Workflow Definitions Definition ID

The DELETE /Workflow/Definitions/{definitionid} method is used to delete the workflow definition with the specified GUID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/modify/

Note: The built-in global workflow definitions (*Global Revocation Workflow* and *Global Enrollment Workflow*) cannot be deleted. A workflow definition cannot be deleted if there is an active or suspended workflow instance for the workflow definition.

Table 796: DELETE Workflow Definitions {definitionid} Input Parameters

Name	In	Description
definitionId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow definition to delete. Use the *GET /Workflow/Definitions* method (see GET Workflow Definitions on page 2542) to retrieve a list of all the workflow definitions to determine the GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.3 GET Workflow Definitions Definition ID

The GET /Workflow/Definitions/{definitionid} method is used to retrieve the workflow definition with the specified GUID. This method returns HTTP 200 OK on a success with details about the specified workflow definition.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/read/

Table 797: GET Workflow Definitions {definitionid} Input Parameters

Name	In	Description
definitionId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow definition to retrieve. Use the *GET /Workflow/Definitions* method (see GET Workflow Definitions on page 2542) to retrieve a list of all the workflow definitions to determine the GUID.
definitionVersion	Query	An integer indicating which version of the workflow definition to return. The default is to return the most recent version (which may not necessarily be the published version).
exportable	Query	A Boolean indicating whether any security RoleIds (see Security Roles on page 2081) in the workflow definition should be removed from the response (true) or not (false). A value of *true* allows for the workflow definition to be exported without role-specific data. The default is *false*.

Table 798: GET Workflow Definitions {definitionsid} Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition.
DisplayName	A string indicating the display name defined for the workflow definition.
Description	A string indicating the description for the workflow definition.
Key	A string indicating the reference key for the workflow definition. The type of information contained in this field will vary depending on the *WorkflowType*. If the *WorkflowType* is *Enrollment* or *Revocation*, this field will contain the Keyfactor Command reference ID for the certificate template. If the *WorkflowType* is *CertificateLeftCollection* or *CertificateEnteredCollection*, this field will contain the Keyfactor Command reference ID for the certificate collection.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template or display name for the certificate collection.
IsPublished	A Boolean indicating whether the workflow definition has been published (true) or not (false). A workflow definition must be published to activate it. For a newly created workflow, this will be *false*.
WorkflowType	A string indicating the type of workflow definition. The currently supported types are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. • **Enrollment (Including Renewals)** The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.

Name	Description
	• **Revocation** The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.
Steps	An array of objects indicating the steps in the workflow definition. The contents of each step will vary depending on the type of workflow and the type of step. For a newly created workflow, there will be no data in this value. Possible steps include:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition step.
DisplayName	A string indicating the display name for the step.
UniqueName	A string indicating the unique name for the step. This value must be unique among the steps in the particular workflow definition. It is intended to be used as a user-friendly reference ID.
ExtensionName	A string indicating the type of step. The currently supported types are: • **Email** Send an email message. This is a separate email message from those typically sent as part of a *Require Approval* step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request. • **PowerShell** Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including: • ConvertFrom-Csv • ConvertFrom-Json • ConvertFrom-Markdown

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>• ConvertFrom-SddlString • ConvertFrom-StringData • ConvertTo-Csv • ConvertTo-Html • ConvertTo-Json • ConvertTo-Xml • ForEach-Object • Get-Command • Where-Object • **CustomPowerShell** Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See Extensions Scripts on page 1704 for adding scripts to the database. • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step can require approval from just one approver or multiple approvers. The workflow will be suspended at this point until the correct number of approvals from users with the correct security roles is received or until one deny is received before continuing to the next step. As part of this step, an email message is sent indicating whether the step was approved or denied—typically to the requester. This step does not include logic to send an email initiating the approval process (letting users know something needs approval). Use an *Email* type step for this. **Important:** Workflows are not supported with CA delegation when they contain steps that require approval. For more information, see the CA configuration Authorization Methods Tab on page 367. **Note:** The users that you send email to initiating the approval process must be members of a security role that is allowed to submit signals (approve/deny) for the workflow in order to approve or deny the request.</td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> Tip: The workflow builder does not include a step to send a notification to the requester of a certificate on an enrollment once the certificate is issued by the CA (as opposed to approved in Keyfactor Command). Use the issued alerts for this (see <a>Issued Request Alert Operations on page 189). - RESTRequest Run a REST (API) request using Active Directory as an identity provider and Basic or Windows authentication. The REST request contents are embedded within the step. It does not call out to an external file. - OAuthRESTRequest Run a REST (API) request using an identity provider other than Active Directory and Token authentication. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent (Enrollment Only) On an enrollment (either CSR or PFX), create a resigned CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) to modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types or a custom step type. The step creates a new CSR using the same public key as the original CSR using the updated SAN and/or subject values. It signs the new CSR with the certificate provided in the step's configuration. For this type of step you will need an enrollment agent certificate available as a PKCS#12 (.PFX) file with included private key to import into Keyfactor Command. This can be a user certificate or a computer certificate (e.g. generated from a copy of the Enrollment Agent template or the Enrollment Agent (Computer) template) and must have a Certificate Request Agent EKU. Note that the built-in Enrollment Agent and Enrollment Agent (Computer) templates do not allow private keys to be exported by default. You will need a template that allows private key export or will need to manually override private key export to create a certificate with an exportable private key in order to create a PKCS#12 (.PFX) file. </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> **Important:** This step applies to Microsoft CAs only. If this step is added to workflow for requests directed to an EJBCA CA, it will fail on enrollment. Note that EJBCA supports submission of updated SAN or subject details as part of standard functionality. - SubjectFormatter (Enrollment Only) On an enrollment done through the Keyfactor Windows Enrollment Gateway using a client-side template configured with the Build from this Active Directory information option on the template, this workflow step handles formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as Build from this Active Directory information must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. **Important:** The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the Keyfactor Windows Enrollment Gateway Installation and Configuration Guide. - EnrollStep Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. - NOOPStep An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. - RevokeStep Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately </td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>following the EndNOOP step.

Tip: For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 2294) and are not configured individually in the workflow steps.</td></tr>
<tr><td>Enabled</td><td>A Boolean indicating whether the step is enabled to run (true) or not (false).</td></tr>
<tr><td>Config-urationPara-meters</td><td>An object containing the configuration parameters for the workflow definition step. These will vary depending on the type of workflow and the type of step (see ExtensionName).

Note: There are no ConfigurationParameters for steps of type SubjectFormatter, EnrollStep, NOOPStep, or RevokeStep.

Possible CustomPowerShell parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>ScriptParameters</td><td>An object defining any parameters to be used in the PowerShell script.</td></tr>
<tr><td>ScriptName</td><td>A string containing the Keyfactor Command reference name of PowerShell script as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
A sample PowerShell script (CustomPower-shellExample.ps1) is provided in the \Exten-sionLibrary\net6.0\Workflow directory on the Keyfactor Command server under the install directory. By default, this is:

<code>C:\Program Files\Keyfactor\Keyfactor Platform \ExtensionLibrary\net6.0\Workfl ow</code></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append additional data to it using PowerShell.

Possible **Email** parameters include:

Value	Description
Subject	A string indicating the subject line for the email message that will be delivered when the workflow definition step is executed.
Message	A string indicating the email message that will be delivered when the workflow definition step is executed. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. For example, for an enrollment pending request notification: “Hello,\n\nA certificate using the $(template) template was requested by $(requester:displayname) from $(CA) on $(subdate). The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: $(request:cn)</td><td>App Owner First Name: $(metadata:ApOwnerFirstName)</td></tr>\n<tr><td>DN: $(request:dn)</td><td>App Owner Last Name: $(metadata:ApOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:ApOwnerEmailAddress)</td></tr>\n<tr><td> </td><td>Business

Name	Description

Name	Description

Value	Description
	Critical: $(metadata:BusinessCritical)</td></tr>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n$(reviewlink)\n\nThanks!\n\nYour Certificate Management Tool\n” See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. **Note:** The $(requester:displayname) substitutable special text token is only supported in environments using Active Directory as an identity provider.
Recipients	An array of strings containing the recipients for the workflow definition email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester). Possible **EnrollmentAgent** parameters include: <table><tr><th>Value</th><th>Description</th></tr><tr><td>EnrollmentAgentCert</td><td>A string containing the base-64-encoded representation of the enrollment agent certificate with private key (in PKCS#12 format) that will be used to sign the CSR. This can be either a user certificate or a computer certificate and must have a Certificate Request Agent EKU.</td></tr><tr><td>EnrollmentAgentCertPassword</td><td>An object indicating the password information used to secure the private key of the enrollment agent certificate. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.</td></tr></table></td></tr></table>

Name	Description

Name	Description

Value	Description
	• Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.

Possible **PowerShell** parameters include:

Value	Description
ScriptParameters	An object defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>additional data to it using PowerShell. Possible **RequireApproval** parameters include:<table><tr><th>Value</th><th>Description</th></tr><tr><td>MinimumApprovals</td><td>In integer indicating the minimum number of users who must approve the request to allow the request to complete.</td></tr><tr><td>DenialEmailSubject</td><td>A string indicating the subject line for the email message that will be delivered if the request is denied.</td></tr><tr><td>DenialEmailMessage</td><td>A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens.</td></tr><tr><td>DenialEmailRecipients</td><td>An array of strings containing the recipients for the denial email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.</td></tr></table></td></tr></table>

Name	Description
	Name / **Description** **Value** / **Description** **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress). ApprovalEmailSubject — A string indicating the subject line for the email message that will be delivered if the request is approved. ApprovalEmailMessage — A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. ApprovalEmailRecipients — An array of strings containing the recipients for the approval email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on

Name	Description

Name	Description

Value	Description
	a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:ApOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header.

Name	Description
	Name / **Description**

Name

Description

Value	Description
	For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBucketProperty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)`

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
UseBasic-Auth	A Boolean indicating whether Basic authentication should be used for the request (True) or not (False). If *UseBasicAuth* is *False*, Windows authentication in the context of the Keyfactor Command application pool user will be used (see Create Service Accounts for Keyfactor Command on page 2757).
BasicUser-name	An object indicating the username information to use for authentication if *UseBasicAuth* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
BasicPass-word	An object indicating the password information to use for authentication if *UseBasicAuth* is *True*. The syntax is the same as for *BasicUsername*.

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	Due to its sensitive nature, this value is not returned in responses.

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example:
	`192.168.12.0/24,192.168.14.22/24`
	When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.

ContentTy-	A string indicating the content type for the request.

Name	Description

Name	Description

Value	Description
pe	Supported values are: • application/json

RequestC-ontent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):
	
	`{`
	`"Id": "$(certid)",`
	`"Metadata":{`
	`"RevocationComment": "$(cmnt)"`
	`}`
	`}`
	
	Note: This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header. For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBuck-etProp-erty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the

Name	Description

Name	Description

Value	Description
	following: `$(MyResponse.[0].ClientMachine)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
client_id	A string indicating the ID of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844).
client_secret	An object indicating the secret of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844). Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned

Name	Description
	Name / **Description**

Value	Description
	in responses.
TokenEndpoint	A string container the URL of the token endpoint for your identity provider. For example: `https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token` For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_

Name	Description

Name	Description

Value	Description
	OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example: `192.168.12.0/24,192.168.14.22/24` When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.
ContentType	A string indicating the content type for the request. Supported values are: • application/json

RequestContent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):
	
	`{`
	`"Id": "$(certid)",`
	`"Metadata":{`
	`"RevocationComment": "$(cmnt)"`
	`}`
	`}`
	
	Note: This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).</td></tr>
<tr><td>Signals</td><td>An array of objects containing data used at the point in the workflow step where the workflow needs to continue based on user input. These will vary depending on the type of workflow and the type of step (see ExtensionName). Possible RequireApproval values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>RoleIds</td><td>An array of integers indicating the security roles whose members are allowed to approve the request.</td></tr>
<tr><td>SignalName</td><td>A string indicating the name of the signal. This value will vary depending on the workflow step. For the built-in Require Approval step, the SignalName is ApprovalStatus.</td></tr>
</table>
Important: If all the security roles configured for a workflow step are deleted from Keyfactor Command, no users will be able to submit signals for workflow instances initiated with that workflow definition. To remedy this, update the workflow definition with one or more current security roles, re-publish it, and then restart any outstanding workflow instances.</td></tr>
<tr><td>Conditions</td><td>An object containing conditions indicating whether the step should run (true) or not (false). Conditions may either have a static value of True or False or a token that will have a value of True or False at the time the step is run. More than one condition may be added. If multiple conditions are used in the same step, all conditions must have a value of True at the time the step is evaluated to be run in order for the step to run. If any single condition evaluates to False, the step will not run. Condition values are:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the condition.</td></tr>
<tr><td>Value</td><td>A string indicating the value of the condition. This should be one of true, false, or a token that will be set to either true or false in an earlier step in the workflow (see <a>Adding, Copying or Modifying a Workflow Definition on page 237 for an example).</td></tr>
</table>
</td></tr>
<tr><td>Outputs</td><td>An object indicating the next step in the workflow. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>continue</td><td>A string indicating the UniqueName of the next workflow step in the chain. This value will be null for the final step in the chain.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>DraftVersion</td><td>An integer indicating the version number of the workflow definition. If this version number does not match the PublishedVersion, changes have been made to the workflow definition that have not yet been published.</td></tr>
<tr><td>PublishedVersion</td><td>An integer indicating the currently published version number of the workflow definition. For a newly created workflow, this value will be null.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.4 PUT Workflow Definitions Definition ID

The PUT /Workflow/Definitions/{definitionid} method is used to update the name and description for the workflow definition with the specified GUID. This method returns HTTP 200 OK on a success with details about the updated workflow definition.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/modify/

Tip: A given workflow can only apply to one key. If you need to run the same workflow steps for more than one key (e.g. the same enrollment steps for more than one template), you can either add these steps to the global workflow or, if you want to run the steps for more than one type of enrollment, for example, but not all, you can configure one custom workflow and then export and re-import that workflow to duplicate it (see PUT Workflow Definitions Definition ID on the previous page) and edit the copy to change the key.

Note: If you edit an existing *published* workflow definition, a new version of the workflow definition will be created. If you edit an existing workflow definition which has *never been published*, the existing configuration will be overwritten with the changes you've made—a new version will not be created.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 799: PUT Workflow Definitions {definitionid} Input Parameters

Name	In	Description
definitionId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow definition.
DisplayName	Body	**Required**. A string indicating the display name defined for the workflow definition.
Description	Body	A string indicating the description for the workflow definition.

Table 800: PUT Workflow Definitions {definitionid} Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition.
DisplayName	A string indicating the display name defined for the workflow definition.
Description	A string indicating the description for the workflow definition.
Key	A string indicating the reference key for the workflow definition. The type of information contained in this field will vary depending on the *WorkflowType*. If the *WorkflowType* is *Enrollment* or *Revocation*, this field will contain the Keyfactor Command reference ID for the certificate template. If the *WorkflowType* is *CertificateLeftCollection* or *CertificateEnteredCollection*, this field will contain the Keyfactor Command reference ID for the certificate collection.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template or display name for the certificate collection.
IsPublished	A Boolean indicating whether the workflow definition has been published (true) or not (false). A workflow definition must be published to activate it. For a newly created workflow, this will be *false*.
WorkflowType	A string indicating the type of workflow definition. The currently supported types are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. • **Enrollment (Including Renewals)** The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.

Name	Description
	• **Revocation** The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.
Steps	An array of objects indicating the steps in the workflow definition. The contents of each step will vary depending on the type of workflow and the type of step. For a newly created workflow, there will be no data in this value. Possible steps include:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition step.
DisplayName	A string indicating the display name for the step.
UniqueName	A string indicating the unique name for the step. This value must be unique among the steps in the particular workflow definition. It is intended to be used as a user-friendly reference ID.
ExtensionName	A string indicating the type of step. The currently supported types are: • **Email** Send an email message. This is a separate email message from those typically sent as part of a *Require Approval* step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request. • **PowerShell** Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including: • ConvertFrom-Csv • ConvertFrom-Json • ConvertFrom-Markdown

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> - ConvertFrom-SddlString - ConvertFrom-StringData - ConvertTo-Csv - ConvertTo-Html - ConvertTo-Json - ConvertTo-Xml - ForEach-Object - Get-Command - Where-Object - CustomPowerShell Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See <a>Extensions Scripts on page 1704 for adding scripts to the database. - RequireApproval Require approval for a workflow step before the step can be completed. The require approval step can require approval from just one approver or multiple approvers. The workflow will be suspended at this point until the correct number of approvals from users with the correct security roles is received or until one deny is received before continuing to the next step. As part of this step, an email message is sent indicating whether the step was approved or denied—typically to the requester. This step does not include logic to send an email initiating the approval process (letting users know something needs approval). Use an Email type step for this. Important: Workflows are not supported with CA delegation when they contain steps that require approval. For more information, see the CA configuration <a>Authorization Methods Tab on page 367. Note: The users that you send email to initiating the approval process must be members of a security role that is allowed to submit signals (approve/deny) for the workflow in order to approve or deny the request. </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> Tip: The workflow builder does not include a step to send a notification to the requester of a certificate on an enrollment once the certificate is issued by the CA (as opposed to approved in Keyfactor Command). Use the issued alerts for this (see <a>Issued Request Alert Operations on page 189). - RESTRequest Run a REST (API) request using Active Directory as an identity provider and Basic or Windows authentication. The REST request contents are embedded within the step. It does not call out to an external file. - OAuthRESTRequest Run a REST (API) request using an identity provider other than Active Directory and Token authentication. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent (Enrollment Only) On an enrollment (either CSR or PFX), create a resigned CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) to modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types or a custom step type. The step creates a new CSR using the same public key as the original CSR using the updated SAN and/or subject values. It signs the new CSR with the certificate provided in the step's configuration. For this type of step you will need an enrollment agent certificate available as a PKCS#12 (.PFX) file with included private key to import into Keyfactor Command. This can be a user certificate or a computer certificate (e.g. generated from a copy of the Enrollment Agent template or the Enrollment Agent (Computer) template) and must have a Certificate Request Agent EKU. Note that the built-in Enrollment Agent and Enrollment Agent (Computer) templates do not allow private keys to be exported by default. You will need a template that allows private key export or will need to manually override private key export to create a certificate with an exportable private key in order to create a PKCS#12 (.PFX) file. </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> **Important:** This step applies to Microsoft CAs only. If this step is added to workflow for requests directed to an EJBCA CA, it will fail on enrollment. Note that EJBCA supports submission of updated SAN or subject details as part of standard functionality. - SubjectFormatter (Enrollment Only) On an enrollment done through the Keyfactor Windows Enrollment Gateway using a client-side template configured with the Build from this Active Directory information option on the template, this workflow step handles formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as Build from this Active Directory information must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. **Important:** The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the Keyfactor Windows Enrollment Gateway Installation and Configuration Guide. - EnrollStep Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. - NOOPStep An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. - RevokeStep Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately </td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>following the EndNOOP step.

Tip: For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 2294) and are not configured individually in the workflow steps.</td></tr>
<tr><td>Enabled</td><td>A Boolean indicating whether the step is enabled to run (true) or not (false).</td></tr>
<tr><td>Config-urationPara-meters</td><td>An object containing the configuration parameters for the workflow definition step. These will vary depending on the type of workflow and the type of step (see ExtensionName).

Note: There are no ConfigurationParameters for steps of type SubjectFormatter, EnrollStep, NOOPStep, or RevokeStep.

Possible CustomPowerShell parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>ScriptParameters</td><td>An object defining any parameters to be used in the PowerShell script.</td></tr>
<tr><td>ScriptName</td><td>A string containing the Keyfactor Command reference name of PowerShell script as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
A sample PowerShell script (CustomPower-shellExample.ps1) is provided in the \Exten-sionLibrary\net6.0\Workflow directory on the Keyfactor Command server under the install directory. By default, this is:

<pre>C:\Program Files\Keyfactor\Keyfactor Platform \ExtensionLibrary\net6.0\Workfl ow</pre></td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append additional data to it using PowerShell.

Possible **Email** parameters include:

Value	Description
Subject	A string indicating the subject line for the email message that will be delivered when the workflow definition step is executed.
Message	A string indicating the email message that will be delivered when the workflow definition step is executed. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. For example, for an enrollment pending request notification: “Hello,\n\nA certificate using the $(template) template was requested by $(requester:displayname) from $(CA) on $(subdate). The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: $(request:cn)</td><td>App Owner First Name: $(metadata:ApOwnerFirstName)</td></tr>\n<tr><td>DN: $(request:dn)</td><td>App Owner Last Name: $(metadata:ApOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:ApOwnerEmailAddress)</td></tr>\n<tr><td> </td><td>Business

Name	Description

Name	Description

Value	Description
	Critical: $(metadata:BusinessCritical)</td></tr>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n$(reviewlink)\n\nThanks!\n\nYour Certificate Management Tool\n” See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. **Note:** The $(requester:displayname) substitutable special text token is only supported in environments using Active Directory as an identity provider.
Recipients	An array of strings containing the recipients for the workflow definition email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from

Name	Description

Name	Description

the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester).

Possible **EnrollmentAgent** parameters include:

Value	Description
EnrollmentAgentCert	A string containing the base-64-encoded representation of the enrollment agent certificate with private key (in PKCS#12 format) that will be used to sign the CSR. This can be either a user certificate or a computer certificate and must have a Certificate Request Agent EKU.
EnrollmentAgentCertPassword	An object indicating the password information used to secure the private key of the enrollment agent certificate. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.

Name	Description

Name	Description

Value	Description
	• Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.

Possible **PowerShell** parameters include:

Value	Description
ScriptParameters	An object defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>additional data to it using PowerShell. Possible **RequireApproval** parameters include: <table><tr><th>Value</th><th>Description</th></tr><tr><td>MinimumApprovals</td><td>In integer indicating the minimum number of users who must approve the request to allow the request to complete.</td></tr><tr><td>DenialEmailSubject</td><td>A string indicating the subject line for the email message that will be delivered if the request is denied.</td></tr><tr><td>DenialEmailMessage</td><td>A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens.</td></tr><tr><td>DenialEmailRecipients</td><td>An array of strings containing the recipients for the denial email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.</td></tr></table></td></tr></table>

Name	Description
	Name / **Description** **Value** / **Description** **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:ApOwnerEmailAddress). ApprovalEmailSubject — A string indicating the subject line for the email message that will be delivered if the request is approved. ApprovalEmailMessage — A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. ApprovalEmailRecipients — An array of strings containing the recipients for the approval email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on

Name	Description

Name	Description

Value	Description
	a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:ApOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header.

Name	Description
	Name / **Description**

Name

Description

Value	Description
	For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBucketProperty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)`

Name	Description
	Name / **Description** **Value** / **Description** Verb — A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE UseBasicAuth — A Boolean indicating whether Basic authentication should be used for the request (True) or not (False). If *UseBasicAuth* is *False*, Windows authentication in the context of the Keyfactor Command application pool user will be used (see Create Service Accounts for Keyfactor Command on page 2757). BasicUsername — An object indicating the username information to use for authentication if *UseBasicAuth* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses. BasicPassword — An object indicating the password information to use for authentication if *UseBasicAuth* is *True*. The syntax is the same as for *BasicUsername*.

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	Due to its sensitive nature, this value is not returned in responses.

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example:
	`192.168.12.0/24,192.168.14.22/24`
	When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.

ContentTy-	A string indicating the content type for the request.

Name	Description

Name	Description

Value	Description
pe	Supported values are: • application/json
RequestC-ontent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request): `{ "Id": "$(certid)", "Metadata":{ "RevocationComment": "$(cmnt)" } }` **Note:** This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header. For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBuck-etProp-erty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the

Name	Description
	Name / **Description**

Name

Description

Value	Description
	following: `$(MyResponse.[0].ClientMachine)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
client_id	A string indicating the ID of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844).
client_secret	An object indicating the secret of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844). Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>in responses.</td></tr>
<tr><td>TokenEnd-point</td><td>A string container the URL of the token endpoint for your identity provider. For example:
<pre>https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token</pre>
For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see <a>Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Install-ation on page 2716).</td></tr>
<tr><td>URL</td><td>A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
<pre>https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL</pre>
Or, with tokens:
<pre>https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)</pre>
Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Value	Description
	OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example: `192.168.12.0/24,192.168.14.22/24` When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.
ContentType	A string indicating the content type for the request. Supported values are: • application/json

RequestContent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):
	
	`{`
	`"Id": "$(certid)",`
	`"Metadata":{`
	`"RevocationComment": "$(cmnt)"`
	`}`
	`}`
	
	Note: This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).</td></tr>
<tr><td>Signals</td><td>An array of objects containing data used at the point in the workflow step where the workflow needs to continue based on user input. These will vary depending on the type of workflow and the type of step (see ExtensionName). Possible RequireApproval values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>RoleIds</td><td>An array of integers indicating the security roles whose members are allowed to approve the request.</td></tr>
<tr><td>SignalName</td><td>A string indicating the name of the signal. This value will vary depending on the workflow step. For the built-in Require Approval step, the SignalName is ApprovalStatus.</td></tr>
</table>
Important: If all the security roles configured for a workflow step are deleted from Keyfactor Command, no users will be able to submit signals for workflow instances initiated with that workflow definition. To remedy this, update the workflow definition with one or more current security roles, re-publish it, and then restart any outstanding workflow instances.</td></tr>
<tr><td>Conditions</td><td>An object containing conditions indicating whether the step should run (true) or not (false). Conditions may either have a static value of True or False or a token that will have a value of True or False at the time the step is run. More than one condition may be added. If multiple conditions are used in the same step, all conditions must have a value of True at the time the step is evaluated to be run in order for the step to run. If any single condition evaluates to False, the step will not run. Condition values are:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the condition.</td></tr>
<tr><td>Value</td><td>A string indicating the value of the condition. This should be one of true, false, or a token that will be set to either true or false in an earlier step in the workflow (see Adding, Copying or Modifying a Workflow Definition on page 237 for an example).</td></tr>
</table>
</td></tr>
<tr><td>Outputs</td><td>An object indicating the next step in the workflow. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>continue</td><td>A string indicating the UniqueName of the next workflow step in the chain. This value will be null for the final step in the chain.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>DraftVersion</td><td>An integer indicating the version number of the workflow definition. If this version number does not match the PublishedVersion, changes have been made to the workflow definition that have not yet been published.</td></tr>
<tr><td>PublishedVersion</td><td>An integer indicating the currently published version number of the workflow definition. For a newly created workflow, this value will be null.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.5 GET Workflow Definitions

The GET /Workflow/Definitions method is used to retrieve the list of workflow definitions. This method returns HTTP 200 OK on a success with high level information about the workflow definitions. Use the GET /Workflow/Definitions/{definitionid} method (see GET Workflow Definitions Definition ID on page 2494) to return details including the workflow steps.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/read/

Table 801: GET Workflow Definitions Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Workflow Definitions Search Feature on page 303. The query fields supported for this endpoint are: • DisplayName • Id • IsPublished (true or false) • WorkflowType (CertificateEnteredCollection, CertificateLeftCollection, Enrollment, or Revocation)
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 802: GET Workflow Definitions Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition.
DisplayName	A string indicating the display name defined for the workflow definition.
Key	A string indicating the reference key for the workflow definition. The type of information contained in this field will vary depending on the *WorkflowType*. If the *WorkflowType* is *Enrollment* or *Revocation*, this field will contain the Keyfactor Command reference ID for the certificate template. If the *WorkflowType* is *CertificateLeftCollection* or *CertificateEnteredCollection*, this field will contain the Keyfactor Command reference ID for the certificate collection.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template or display name for the certificate collection.
WorkflowType	A string indicating the type of workflow definition. The currently supported types are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. • **Enrollment (Including Renewals)** The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request. • **Revocation** The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.

Name	Description
DraftVersion	An integer indicating the version number of the workflow definition. If this version number does not match the *PublishedVersion*, changes have been made to the workflow definition that have not yet been published.
PublishedVersion	An integer indicating the currently published version number of the workflow definition.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.6 POST Workflow Definitions

The POST /Workflow/Definitions method is used to create a new workflow definition without any steps. To add steps to the workflow, use the PUT /Workflow/Definitions/{definitionId}/Steps method (see PUT Workflow Definitions Definition ID Steps on page 2579). This method returns HTTP 200 OK on a success with details about the workflow definition.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/modify/

Tip: A given workflow can only apply to one key. If you need to run the same workflow steps for more than one key (e.g. the same enrollment steps for more than one template), you can either add these steps to the global workflow or, if you want to run the steps for more than one type of enrollment, for example, but not all, you can configure one custom workflow and then export and re-import that workflow to duplicate it (see POST Workflow Definitions above) and edit the copy to change the key.

Table 803: POST Workflow Definitions Input Parameters

Name	In	Description
DisplayName	Body	**Required**. A string indicating the display name defined for the workflow definition.
Description	Body	A string indicating the description for the workflow definition.
Key	Body	**Required**. A string indicating the reference key for the workflow definition. The type of information contained in this field will vary depending on the *WorkflowType*. If the *WorkflowType* is *Enrollment* or *Revocation*, this field will contain the Keyfactor Command reference ID for the certificate template. If the *WorkflowType* is *CertificateLeftCollection* or *CertificateEnteredCollection*, this field will contain the Keyfactor Command reference ID for the certificate collection. Use the GET /Templates method (see GET Templates on page 2422) to retrieve a list or your certificate templates to determine the template ID. Use the GET /CertificateCollections method (see GET Certificate Collections on page 1296) to retrieve a list or your certificate collections to determine the collection ID. This field cannot be modified on an edit.
KeyDisplayName	Body	A string indicating the friendly name defined in Keyfactor Command for the certificate template or display name for the certificate collection.
WorkflowType	Body	**Required**. A string indicating the type of workflow definition. The currently supported types are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a

Name	In	Description
		support ticket request to investigate the removal of a web server certificate. • **Enrollment (Including Renewals)** The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request. • **Revocation** The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field. This field cannot be modified on an edit.

Table 804: POST Workflow Definitions Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition.
DisplayName	A string indicating the display name defined for the workflow definition.
Description	A string indicating the description for the workflow definition.
Key	A string indicating the reference key for the workflow definition. The type of information contained in this field will vary depending on the *WorkflowType*. If the *WorkflowType* is *Enrollment* or *Revocation*, this field will contain the Keyfactor Command reference ID for the certificate template. If the *WorkflowType* is *CertificateLeftCollection* or *CertificateEnteredCollection*, this field will contain the Keyfactor Command reference ID for the certificate collection.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template or display name for the certificate collection.
IsPublished	A Boolean indicating whether the workflow definition has been published (true) or not (false). A workflow definition must be published to activate it. For a newly created workflow, this will be *false*.
WorkflowType	A string indicating the type of workflow definition. The currently supported types are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. • **Enrollment (Including Renewals)** The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.

Name	Description
	• **Revocation** The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.
Steps	An array of objects indicating the steps in the workflow definition. The contents of each step will vary depending on the type of workflow and the type of step. For a newly created workflow, there will be no data in this value. Possible steps include:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition step.
DisplayName	A string indicating the display name for the step.
UniqueName	A string indicating the unique name for the step. This value must be unique among the steps in the particular workflow definition. It is intended to be used as a user-friendly reference ID.
ExtensionName	A string indicating the type of step. The currently supported types are: • **Email** Send an email message. This is a separate email message from those typically sent as part of a *Require Approval* step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request. • **PowerShell** Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including: • ConvertFrom-Csv • ConvertFrom-Json • ConvertFrom-Markdown

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> - ConvertFrom-SddlString - ConvertFrom-StringData - ConvertTo-Csv - ConvertTo-Html - ConvertTo-Json - ConvertTo-Xml - ForEach-Object - Get-Command - Where-Object - CustomPowerShell Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See <a>Extensions Scripts on page 1704 for adding scripts to the database. - RequireApproval Require approval for a workflow step before the step can be completed. The require approval step can require approval from just one approver or multiple approvers. The workflow will be suspended at this point until the correct number of approvals from users with the correct security roles is received or until one deny is received before continuing to the next step. As part of this step, an email message is sent indicating whether the step was approved or denied—typically to the requester. This step does not include logic to send an email initiating the approval process (letting users know something needs approval). Use an Email type step for this. Important: Workflows are not supported with CA delegation when they contain steps that require approval. For more information, see the CA configuration <a>Authorization Methods Tab on page 367. Note: The users that you send email to initiating the approval process must be members of a security role that is allowed to submit signals (approve/deny) for the workflow in order to approve or deny the request. </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> Tip: The workflow builder does not include a step to send a notification to the requester of a certificate on an enrollment once the certificate is issued by the CA (as opposed to approved in Keyfactor Command). Use the issued alerts for this (see <a>Issued Request Alert Operations on page 189). - RESTRequest Run a REST (API) request using Active Directory as an identity provider and Basic or Windows authentication. The REST request contents are embedded within the step. It does not call out to an external file. - OAuthRESTRequest Run a REST (API) request using an identity provider other than Active Directory and Token authentication. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent (Enrollment Only) On an enrollment (either CSR or PFX), create a resigned CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) to modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types or a custom step type. The step creates a new CSR using the same public key as the original CSR using the updated SAN and/or subject values. It signs the new CSR with the certificate provided in the step's configuration. For this type of step you will need an enrollment agent certificate available as a PKCS#12 (.PFX) file with included private key to import into Keyfactor Command. This can be a user certificate or a computer certificate (e.g. generated from a copy of the Enrollment Agent template or the Enrollment Agent (Computer) template) and must have a Certificate Request Agent EKU. Note that the built-in Enrollment Agent and Enrollment Agent (Computer) templates do not allow private keys to be exported by default. You will need a template that allows private key export or will need to manually override private key export to create a certificate with an exportable private key in order to create a PKCS#12 (.PFX) file. </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> **Important:** This step applies to Microsoft CAs only. If this step is added to workflow for requests directed to an EJBCA CA, it will fail on enrollment. Note that EJBCA supports submission of updated SAN or subject details as part of standard functionality. - SubjectFormatter (Enrollment Only) On an enrollment done through the Keyfactor Windows Enrollment Gateway using a client-side template configured with the Build from this Active Directory information option on the template, this workflow step handles formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as Build from this Active Directory information must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. Important: The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the Keyfactor Windows Enrollment Gateway Installation and Configuration Guide. - EnrollStep Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. - NOOPStep An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. - RevokeStep Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately </td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>following the EndNOOP step.

Tip: For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 2294) and are not configured individually in the workflow steps.</td></tr>
<tr><td>Enabled</td><td>A Boolean indicating whether the step is enabled to run (true) or not (false).</td></tr>
<tr><td>Config-urationPara-meters</td><td>An object containing the configuration parameters for the workflow definition step. These will vary depending on the type of workflow and the type of step (see ExtensionName).

Note: There are no ConfigurationParameters for steps of type SubjectFormatter, EnrollStep, NOOPStep, or RevokeStep.

Possible CustomPowerShell parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>ScriptParameters</td><td>An object defining any parameters to be used in the PowerShell script.</td></tr>
<tr><td>ScriptName</td><td>A string containing the Keyfactor Command reference name of PowerShell script as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
A sample PowerShell script (CustomPower-shellExample.ps1) is provided in the \Exten-sionLibrary\net6.0\Workflow directory on the Keyfactor Command server under the install directory. By default, this is:

<pre>C:\Program Files\Keyfactor\Keyfactor Platform \ExtensionLibrary\net6.0\Workfl ow</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append additional data to it using PowerShell.

Possible **Email** parameters include:

Value	Description
Subject	A string indicating the subject line for the email message that will be delivered when the workflow definition step is executed.
Message	A string indicating the email message that will be delivered when the workflow definition step is executed. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. For example, for an enrollment pending request notification: “Hello,\n\nA certificate using the $(template) template was requested by $(requester:displayname) from $(CA) on $(subdate). The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: $(request:cn)</td><td>App Owner First Name: $(metadata:ApOwnerFirstName)</td></tr>\n<tr><td>DN: $(request:dn)</td><td>App Owner Last Name: $(metadata:ApOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:ApOwnerEmailAddress)</td></tr>\n<tr><td> </td><td>Business

Name	Description

Name	Description

Value	Description
	Critical: $(metadata:BusinessCritical)</td></tr>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n$(reviewlink)\n\nThanks!\n\nYour Certificate Management Tool\n” See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. **Note:** The $(requester:displayname) substitutable special text token is only supported in environments using Active Directory as an identity provider.
Recipients	An array of strings containing the recipients for the workflow definition email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester). Possible **EnrollmentAgent** parameters include: <table><tr><th>Value</th><th>Description</th></tr><tr><td>EnrollmentAgentCert</td><td>A string containing the base-64-encoded representation of the enrollment agent certificate with private key (in PKCS#12 format) that will be used to sign the CSR. This can be either a user certificate or a computer certificate and must have a Certificate Request Agent EKU.</td></tr><tr><td>EnrollmentAgentCertPassword</td><td>An object indicating the password information used to secure the private key of the enrollment agent certificate. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.</td></tr></table></td></tr></table>

Name	Description

Name	Description

Value	Description
	• Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.

Possible **PowerShell** parameters include:

Value	Description
ScriptParameters	An object defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append

Name	Description
	Name / **Description**

additional data to it using PowerShell.

Possible **RequireApproval** parameters include:

Value	Description
MinimumApprovals	In integer indicating the minimum number of users who must approve the request to allow the request to complete.
DenialEmailSubject	A string indicating the subject line for the email message that will be delivered if the request is denied.
DenialEmailMessage	A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens.
DenialEmailRecipients	An array of strings containing the recipients for the denial email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.

Name	Description
	Name / **Description** **Value** / **Description** **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress). ApprovalEmailSubject — A string indicating the subject line for the email message that will be delivered if the request is approved. ApprovalEmailMessage — A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. ApprovalEmailRecipients — An array of strings containing the recipients for the approval email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on

Name	Description

Name	Description

Value	Description
	a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:ApOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header.

Name	Description
	Name / **Description**

Name

Description

Value	Description
	For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBucketProperty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)`

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
UseBasic-Auth	A Boolean indicating whether Basic authentication should be used for the request (True) or not (False). If *UseBasicAuth* is *False*, Windows authentication in the context of the Keyfactor Command application pool user will be used (see Create Service Accounts for Keyfactor Command on page 2757).
BasicUser-name	An object indicating the username information to use for authentication if *UseBasicAuth* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
BasicPass-word	An object indicating the password information to use for authentication if *UseBasicAuth* is *True*. The syntax is the same as for *BasicUsername*.

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	Due to its sensitive nature, this value is not returned in responses.

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example:
	`192.168.12.0/24,192.168.14.22/24`
	When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.

ContentTy-	A string indicating the content type for the request.

Name	Description

Name	Description

Value	Description
pe	Supported values are: • application/json
RequestC-ontent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request): `{ "Id": "$(certid)", "Metadata":{ "RevocationComment": "$(cmnt)" } }` **Note:** This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header. For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBuck-etProp-erty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the

Name	Description
	Name / **Description**

Name

Description

Value	Description
	following: `$(MyResponse.[0].ClientMachine)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
client_id	A string indicating the ID of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844).
client_secret	An object indicating the secret of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844). Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned

Name	Description
	Name / **Description**

Value	Description
	in responses.
TokenEndpoint	A string container the URL of the token endpoint for your identity provider. For example: `https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token` For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_

Name	Description

Name	Description

Value	Description
	OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example: `192.168.12.0/24,192.168.14.22/24` When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.
ContentType	A string indicating the content type for the request. Supported values are: • application/json

RequestContent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):
	
	`{`
	`"Id": "$(certid)",`
	`"Metadata":{`
	`"RevocationComment": "$(cmnt)"`
	`}`
	`}`
	
	Note: This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).</td></tr>
<tr><td>Signals</td><td>An array of objects containing data used at the point in the workflow step where the workflow needs to continue based on user input. These will vary depending on the type of workflow and the type of step (see ExtensionName). Possible RequireApproval values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>RoleIds</td><td>An array of integers indicating the security roles whose members are allowed to approve the request.</td></tr>
<tr><td>SignalName</td><td>A string indicating the name of the signal. This value will vary depending on the workflow step. For the built-in Require Approval step, the SignalName is ApprovalStatus.</td></tr>
</table>
Important: If all the security roles configured for a workflow step are deleted from Keyfactor Command, no users will be able to submit signals for workflow instances initiated with that workflow definition. To remedy this, update the workflow definition with one or more current security roles, re-publish it, and then restart any outstanding workflow instances.</td></tr>
<tr><td>Conditions</td><td>An object containing conditions indicating whether the step should run (true) or not (false). Conditions may either have a static value of True or False or a token that will have a value of True or False at the time the step is run. More than one condition may be added. If multiple conditions are used in the same step, all conditions must have a value of True at the time the step is evaluated to be run in order for the step to run. If any single condition evaluates to False, the step will not run. Condition values are:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the condition.</td></tr>
<tr><td>Value</td><td>A string indicating the value of the condition. This should be one of true, false, or a token that will be set to either true or false in an earlier step in the workflow (see <a>Adding, Copying or Modifying a Workflow Definition on page 237 for an example).</td></tr>
</table>
</td></tr>
<tr><td>Outputs</td><td>An object indicating the next step in the workflow. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>continue</td><td>A string indicating the UniqueName of the next workflow step in the chain. This value will be null for the final step in the chain.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>DraftVersion</td><td>An integer indicating the version number of the workflow definition. If this version number does not match the PublishedVersion, changes have been made to the workflow definition that have not yet been published.</td></tr>
<tr><td>PublishedVersion</td><td>An integer indicating the currently published version number of the workflow definition. For a newly created workflow, this value will be null.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.7 GET Workflow Definitions Steps

The GET /Workflow/Definitions/Steps method is used to retrieve the workflow definition step structure for the workflow definition steps. This method returns HTTP 200 OK on a success with information about the structure of the workflow definition steps.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/read/

Table 805: GET Workflow Definitions Steps Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Workflow Definitions Search Feature on page 303. The query fields supported for this endpoint are: • DisplayName • ExtensionName • SupportedWorkflowTypes
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *DisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 806: GET Workflow Definitions Steps Response Data

Name	Description
DisplayName	A string indicating the display name of the workflow definition step.
ExtensionName	A string indicating the extension name of the workflow definition step. The built-in extension names are: • **Email** Send an email message. This is a separate email message from those typically sent as part of a *Require Approval* step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request. • **PowerShell** Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including: • ConvertFrom-Csv • ConvertFrom-Json • ConvertFrom-Markdown • ConvertFrom-SddlString • ConvertFrom-StringData • ConvertTo-Csv • ConvertTo-Html • ConvertTo-Json • ConvertTo-Xml • ForEach-Object • Get-Command • Where-Object • **CustomPowerShell** Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See Extensions Scripts on page 1704 for

Name	Description
	adding scripts to the database. • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step can require approval from just one approver or multiple approvers. The workflow will be suspended at this point until the correct number of approvals from users with the correct security roles is received or until one deny is received before continuing to the next step. As part of this step, an email message is sent indicating whether the step was approved or denied—typically to the requester. This step does not include logic to send an email initiating the approval process (letting users know something needs approval). Use an *Email* type step for this. **Important:** Workflows are not supported with CA delegation when they contain steps that require approval. For more information, see the CA configuration Authorization Methods Tab on page 367. **Note:** The users that you send email to initiating the approval process must be members of a security role that is allowed to submit signals (approve/deny) for the workflow in order to approve or deny the request. **Tip:** The workflow builder does not include a step to send a notification to the requester of a certificate on an enrollment once the certificate is issued by the CA (as opposed to approved in Keyfactor Command). Use the issued alerts for this (see Issued Request Alert Operations on page 189). • **RESTRequest** Run a REST (API) request using Active Directory as an identity provider and Basic or Windows authentication. The REST request contents are embedded within the step. It does not call out to an external file. • **OAuthRESTRequest** Run a REST (API) request using an identity provider other than Active Directory and Token authentication. The REST request contents are embedded within the step. It does not call out to an external file.

Name	Description
	• **EnrollmentAgent (Enrollment Only)** On an enrollment (either CSR or PFX), create a resigned CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) to modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types or a custom step type. The step creates a new CSR using the same public key as the original CSR using the updated SAN and/or subject values. It signs the new CSR with the certificate provided in the step's configuration. For this type of step you will need an enrollment agent certificate available as a PKCS#12 (.PFX) file with included private key to import into Keyfactor Command. This can be a user certificate or a computer certificate (e.g. generated from a copy of the *Enrollment Agent* template or the *Enrollment Agent (Computer)* template) and must have a Certificate Request Agent EKU. Note that the built-in *Enrollment Agent* and *Enrollment Agent (Computer)* templates do not allow private keys to be exported by default. You will need a template that allows private key export or will need to manually override private key export to create a certificate with an exportable private key in order to create a PKCS#12 (.PFX) file. ⚠ **Important:** This step applies to Microsoft CAs only. If this step is added to workflow for requests directed to an EJBCA CA, it will fail on enrollment. Note that EJBCA supports submission of updated SAN or subject details as part of standard functionality. • **SubjectFormatter (Enrollment Only)** On an enrollment done through the Keyfactor Windows Enrollment Gateway using a client-side template configured with the *Build from this Active Directory information* option on the template, this workflow step handles formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template

Name	Description
	configured with the subject as *Build from this Active Directory information* must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. **Important:** The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the *Keyfactor Windows Enrollment Gateway Installation and Configuration Guide*. • **EnrollStep** Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. • **NOOPStep** An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. • **RevokeStep** Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately following the EndNOOP step.
SupportedWorkflowTypes	An array of strings containing a list of the workflow types supported by the workflow definition step. Possible built-in values are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor

Name	Description
	Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. • **Enrollment (Including Renewals)** The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request. • **Revocation** The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.
ConfigurationParametersDefinition	An object containing the configuration parameters for the workflow definition step. These will vary depending on the step.
SignalsDefinition	An object containing the signals defined for the workflow definition step. These will vary depending on the step.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.8 GET Workflow Definitions Types

The GET /Workflow/Definitions/Types method is used to retrieve the workflow definition types that have been defined for use. This method returns HTTP 200 OK on a success with information about the defined workflow definition types.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/read/

Table 807: GET Workflow Definitions Types Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Workflow Definitions Search Feature on page 303. The query fields supported for this endpoint are: • Name
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *WorkflowType*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 808: GET Workflow Definitions Types Response Data

Name	Description
WorkflowType	A string indicating the display name of the workflow type.
KeyType	A string indicating the key type for the workflow. The built-in enrollment and revocation workflows use *Templates* as the key type. The built-in certificate entered collection and certificate left collection workflows use *Certificate Collections* as the key type.
ContextParameters	An object containing the tokens that the workflow type provider has the ability to replace. These will vary depending on the workflow type.
BuiltInSteps	An object containing the information about the built-in step(s) for the workflow type (e.g. the enrollment step of the enrollment type). Possible steps include:

Name	Description
DisplayName	A string indicating the display name for the step.
ExtensionName	A string indicating the extension name for the step. The built-in extensions are: • EnrollStep • RevokeStep
Outputs	An array of strings containing the outputs for the workflow definition step. For the built-in steps, the only output is an indicator for the next step in the workflow or that the workflow is complete.
ConfigurationParametersDefinition	An object containing the configuration parameters for the workflow definition step. These will vary depending on the step.
SignalsDefinition	An object containing the signals defined for the workflow definition step. These will vary depending on the step.

Name	Description
	Note: There are no built-in steps for workflows of types *certificate entered collection* and *certificate left collection*.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.9 PUT Workflow Definitions Definition ID Steps

The PUT /Workflow/Definitions/{definitionid}/Steps method is used to add or update the workflow steps for the workflow definition with the specified GUID. This method returns HTTP 200 OK on a success with details about the updated workflow definition.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/modify/

Note: If you edit an existing *published* workflow definition, a new version of the workflow definition will be created. If you edit an existing workflow definition which has *never been published*, the existing configuration will be overwritten with the changes you've made—a new version will not be created.

Important: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 809: PUT Workflow Definitions {definitionid} Steps Input Parameters

Name	In	Description
definitionId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow definition to update. Use the *GET /Workflow/Definitions* method (see GET Workflow Definitions on page 2542) to retrieve a list of all the workflow definitions to determine the GUID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>request</td><td>Body</td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>DisplayName</td><td>A string indicating the display name for the step.</td></tr>
<tr><td>UniqueName</td><td>A string indicating the unique name for the step. This value must be unique among the steps in the particular workflow definition. It is intended to be used as a user-friendly reference ID.</td></tr>
<tr><td>ExtensionName</td><td>A string indicating the type of step. The currently supported types are:
Email
Send an email message. This is a separate email message from those typically sent as part of a Require Approval step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request.
PowerShell
Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including:
ConvertFrom-Csv
ConvertFrom-Json
ConvertFrom-Markdown
ConvertFrom-SddlString
ConvertFrom-StringData
ConvertTo-Csv
ConvertTo-Html
ConvertTo-Json
ConvertTo-Xml
ForEach-Object
Get-Command
Where-Object
CustomPowerShell
Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See <a>Extensions Scripts on page 1704 for adding scripts to the database.
RequireApproval
Require approval for a workflow step before the step can be completed. The require approval step can require approval from</td></tr>
</table>
</td></tr>
</table>

Table 810: PUT Workflow Definitions {definitionid} Steps Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition.
DisplayName	A string indicating the display name defined for the workflow definition.
Description	A string indicating the description for the workflow definition.
Key	A string indicating the reference key for the workflow definition. The type of information contained in this field will vary depending on the *WorkflowType*. If the *WorkflowType* is *Enrollment* or *Revocation*, this field will contain the Keyfactor Command reference ID for the certificate template. If the *WorkflowType* is *CertificateLeftCollection* or *CertificateEnteredCollection*, this field will contain the Keyfactor Command reference ID for the certificate collection.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template or display name for the certificate collection.
IsPublished	A Boolean indicating whether the workflow definition has been published (true) or not (false). A workflow definition must be published to activate it. For a newly created workflow, this will be *false*.
WorkflowType	A string indicating the type of workflow definition. The currently supported types are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. • **Enrollment (Including Renewals)** The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

- **Revocation**

  The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.

</td></tr>
<tr><td>Steps</td><td>

An array of objects indicating the steps in the workflow definition. The contents of each step will vary depending on the type of workflow and the type of step. For a newly created workflow, there will be no data in this value. Possible steps include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the workflow definition step.</td></tr>
<tr><td>DisplayName</td><td>A string indicating the display name for the step.</td></tr>
<tr><td>UniqueName</td><td>A string indicating the unique name for the step. This value must be unique among the steps in the particular workflow definition. It is intended to be used as a user-friendly reference ID.</td></tr>
<tr><td>ExtensionName</td><td>A string indicating the type of step. The currently supported types are:

- **Email**

  Send an email message. This is a separate email message from those typically sent as part of a *Require Approval* step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request.

- **PowerShell**

  Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including:

  - ConvertFrom-Csv
  - ConvertFrom-Json
  - ConvertFrom-Markdown</td></tr>
</table>

</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>• ConvertFrom-SddlString • ConvertFrom-StringData • ConvertTo-Csv • ConvertTo-Html • ConvertTo-Json • ConvertTo-Xml • ForEach-Object • Get-Command • Where-Object • **CustomPowerShell** Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See Extensions Scripts on page 1704 for adding scripts to the database. • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step can require approval from just one approver or multiple approvers. The workflow will be suspended at this point until the correct number of approvals from users with the correct security roles is received or until one deny is received before continuing to the next step. As part of this step, an email message is sent indicating whether the step was approved or denied—typically to the requester. This step does not include logic to send an email initiating the approval process (letting users know something needs approval). Use an *Email* type step for this. **Important:** Workflows are not supported with CA delegation when they contain steps that require approval. For more information, see the CA configuration Authorization Methods Tab on page 367. **Note:** The users that you send email to initiating the approval process must be members of a security role that is allowed to submit signals (approve/deny) for the workflow in order to approve or deny the request.</td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> Tip: The workflow builder does not include a step to send a notification to the requester of a certificate on an enrollment once the certificate is issued by the CA (as opposed to approved in Keyfactor Command). Use the issued alerts for this (see <a>Issued Request Alert Operations on page 189). - RESTRequest Run a REST (API) request using Active Directory as an identity provider and Basic or Windows authentication. The REST request contents are embedded within the step. It does not call out to an external file. - OAuthRESTRequest Run a REST (API) request using an identity provider other than Active Directory and Token authentication. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent (Enrollment Only) On an enrollment (either CSR or PFX), create a resigned CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) to modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types or a custom step type. The step creates a new CSR using the same public key as the original CSR using the updated SAN and/or subject values. It signs the new CSR with the certificate provided in the step's configuration. For this type of step you will need an enrollment agent certificate available as a PKCS#12 (.PFX) file with included private key to import into Keyfactor Command. This can be a user certificate or a computer certificate (e.g. generated from a copy of the Enrollment Agent template or the Enrollment Agent (Computer) template) and must have a Certificate Request Agent EKU. Note that the built-in Enrollment Agent and Enrollment Agent (Computer) templates do not allow private keys to be exported by default. You will need a template that allows private key export or will need to manually override private key export to create a certificate with an exportable private key in order to create a PKCS#12 (.PFX) file. </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> **Important:** This step applies to Microsoft CAs only. If this step is added to workflow for requests directed to an EJBCA CA, it will fail on enrollment. Note that EJBCA supports submission of updated SAN or subject details as part of standard functionality. - SubjectFormatter (Enrollment Only) On an enrollment done through the Keyfactor Windows Enrollment Gateway using a client-side template configured with the Build from this Active Directory information option on the template, this workflow step handles formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as Build from this Active Directory information must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. Important: The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the Keyfactor Windows Enrollment Gateway Installation and Configuration Guide. - EnrollStep Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. - NOOPStep An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. - RevokeStep Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately </td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>following the EndNOOP step.

Tip: For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 2294) and are not configured individually in the workflow steps.</td></tr>
<tr><td>Enabled</td><td>A Boolean indicating whether the step is enabled to run (true) or not (false).</td></tr>
<tr><td>Config-urationPara-meters</td><td>An object containing the configuration parameters for the workflow definition step. These will vary depending on the type of workflow and the type of step (see ExtensionName).

Note: There are no ConfigurationParameters for steps of type SubjectFormatter, EnrollStep, NOOPStep, or RevokeStep.

Possible CustomPowerShell parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>ScriptParameters</td><td>An object defining any parameters to be used in the PowerShell script.</td></tr>
<tr><td>ScriptName</td><td>A string containing the Keyfactor Command reference name of PowerShell script as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
A sample PowerShell script (CustomPower-shellExample.ps1) is provided in the \Exten-sionLibrary\net6.0\Workflow directory on the Keyfactor Command server under the install directory. By default, this is:

<pre>C:\Program Files\Keyfactor\Keyfactor Platform \ExtensionLibrary\net6.0\Workfl ow</pre></td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append additional data to it using PowerShell.

Possible **Email** parameters include:

Value	Description
Subject	A string indicating the subject line for the email message that will be delivered when the workflow definition step is executed.
Message	A string indicating the email message that will be delivered when the workflow definition step is executed. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. For example, for an enrollment pending request notification: "Hello,\n\nA certificate using the $(template) template was requested by $(requester:displayname) from $(CA) on $(subdate). The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: $(request:cn)</td><td>App Owner First Name: $(metadata:ApOwnerFirstName)</td></tr>\n<tr><td>DN: $(request:dn)</td><td>App Owner Last Name: $(metadata:ApOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:ApOwnerEmailAddress)</td></tr>\n<tr><td> </td><td>Business

Name	Description

Name	Description

Value	Description
	Critical: $(metadata:BusinessCritical)</td></tr>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n$(reviewlink)\n\nThanks!\n\nYour Certificate Management Tool\n” See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. **Note:** The $(requester:displayname) substitutable special text token is only supported in environments using Active Directory as an identity provider.
Recipients	An array of strings containing the recipients for the workflow definition email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from

Name	Description

Name	Description

the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester).

Possible **EnrollmentAgent** parameters include:

Value	Description
EnrollmentAgentCert	A string containing the base-64-encoded representation of the enrollment agent certificate with private key (in PKCS#12 format) that will be used to sign the CSR. This can be either a user certificate or a computer certificate and must have a Certificate Request Agent EKU.
EnrollmentAgentCertPassword	An object indicating the password information used to secure the private key of the enrollment agent certificate. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.

Name	Description

Name	Description

Value	Description
	• Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.

Possible **PowerShell** parameters include:

Value	Description
ScriptParameters	An object defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>additional data to it using PowerShell. Possible **RequireApproval** parameters include: <table><tr><th>Value</th><th>Description</th></tr><tr><td>MinimumApprovals</td><td>In integer indicating the minimum number of users who must approve the request to allow the request to complete.</td></tr><tr><td>DenialEmailSubject</td><td>A string indicating the subject line for the email message that will be delivered if the request is denied.</td></tr><tr><td>DenialEmailMessage</td><td>A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens.</td></tr><tr><td>DenialEmailRecipients</td><td>An array of strings containing the recipients for the denial email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.</td></tr></table></td></tr></table>

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Note: The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:ApOwnerEmailAddress).
ApprovalEmailSubject	A string indicating the subject line for the email message that will be delivered if the request is approved.
ApprovalEmailMessage	A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens.
ApprovalEmailRecipients	An array of strings containing the recipients for the approval email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on

Name	Description

Name	Description

Value	Description
	a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:ApOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header.

Name	Description
	Name / **Description**

Name

Description

Value	Description
	For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBucketProperty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)`

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
UseBasic-Auth	A Boolean indicating whether Basic authentication should be used for the request (True) or not (False). If *UseBasicAuth* is *False*, Windows authentication in the context of the Keyfactor Command application pool user will be used (see Create Service Accounts for Keyfactor Command on page 2757).
BasicUser-name	An object indicating the username information to use for authentication if *UseBasicAuth* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
BasicPass-word	An object indicating the password information to use for authentication if *UseBasicAuth* is *True*. The syntax is the same as for *BasicUsername*.

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	Due to its sensitive nature, this value is not returned in responses.

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example:
	`192.168.12.0/24,192.168.14.22/24`
	When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.

ContentTy-	A string indicating the content type for the request.

Name	Description

Name	Description

Value	Description
pe	Supported values are: • application/json

RequestC-ontent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):
	
	`{`
	`"Id": "$(certid)",`
	`"Metadata":{`
	`"RevocationComment": "$(cmnt)"`
	`}`
	`}`
	
	Note: This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header. For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBuck-etProp-erty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the

Name	Description
	Name / **Description**

Name

Description

Value	Description
	following: `$(MyResponse.[0].ClientMachine)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
client_id	A string indicating the ID of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844).
client_secret	An object indicating the secret of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844). Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned

Name	Description
	Name / **Description**

Name

Description

Value	Description
	in responses.
TokenEndpoint	A string container the URL of the token endpoint for your identity provider. For example: `https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token` For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_

Name	Description

Name	Description

Value	Description
	OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example: `192.168.12.0/24,192.168.14.22/24` When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.
ContentType	A string indicating the content type for the request. Supported values are: • application/json

RequestContent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):
	
	`{`
	`"Id": "$(certid)",`
	`"Metadata":{`
	`"RevocationComment": "$(cmnt)"`
	`}`
	`}`
	
	Note: This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).</td></tr>
<tr><td>Signals</td><td>An array of objects containing data used at the point in the workflow step where the workflow needs to continue based on user input. These will vary depending on the type of workflow and the type of step (see ExtensionName). Possible RequireApproval values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>RoleIds</td><td>An array of integers indicating the security roles whose members are allowed to approve the request.</td></tr>
<tr><td>SignalName</td><td>A string indicating the name of the signal. This value will vary depending on the workflow step. For the built-in Require Approval step, the SignalName is ApprovalStatus.</td></tr>
</table>
Important: If all the security roles configured for a workflow step are deleted from Keyfactor Command, no users will be able to submit signals for workflow instances initiated with that workflow definition. To remedy this, update the workflow definition with one or more current security roles, re-publish it, and then restart any outstanding workflow instances.</td></tr>
<tr><td>Conditions</td><td>An object containing conditions indicating whether the step should run (true) or not (false). Conditions may either have a static value of True or False or a token that will have a value of True or False at the time the step is run. More than one condition may be added. If multiple conditions are used in the same step, all conditions must have a value of True at the time the step is evaluated to be run in order for the step to run. If any single condition evaluates to False, the step will not run. Condition values are:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the condition.</td></tr>
<tr><td>Value</td><td>A string indicating the value of the condition. This should be one of true, false, or a token that will be set to either true or false in an earlier step in the workflow (see <a>Adding, Copying or Modifying a Workflow Definition on page 237 for an example).</td></tr>
</table>
</td></tr>
<tr><td>Outputs</td><td>An object indicating the next step in the workflow. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>continue</td><td>A string indicating the UniqueName of the next workflow step in the chain. This value will be null for the final step in the chain.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>DraftVersion</td><td>An integer indicating the version number of the workflow definition. If this version number does not match the PublishedVersion, changes have been made to the workflow definition that have not yet been published.</td></tr>
<tr><td>PublishedVersion</td><td>An integer indicating the currently published version number of the workflow definition. For a newly created workflow, this value will be null.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.40.10 POST Workflow Definitions Definition ID Publish

The POST /Workflow/Definitions/{definitionid}/Publish method is used to mark the most recent version of the workflow definition with the specified GUID as the published, active, version. When a definition is published, all new or restarted workflow instances (see <a>Workflow Instances on page 2628) will be able to use the updated version of the workflow. This method returns HTTP 200 OK on a success with details about the workflow definition.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/definitions/modify/

Table 811: POST Workflow Definitions {definitionid} Publish Input Parameters

Name	In	Description
definitionId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow definition to publish. Use the *GET /Workflow/Definitions* method (see GET Workflow Definitions on page 2542) to retrieve a list of all the workflow definitions to determine the GUID.

Table 812: POST Workflow Definitions {definitionid} Publish Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition.
DisplayName	A string indicating the display name defined for the workflow definition.
Description	A string indicating the description for the workflow definition.
Key	A string indicating the reference key for the workflow definition. The type of information contained in this field will vary depending on the *WorkflowType*. If the *WorkflowType* is *Enrollment* or *Revocation*, this field will contain the Keyfactor Command reference ID for the certificate template. If the *WorkflowType* is *CertificateLeftCollection* or *CertificateEnteredCollection*, this field will contain the Keyfactor Command reference ID for the certificate collection.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template or display name for the certificate collection.
IsPublished	A Boolean indicating whether the workflow definition has been published (true) or not (false). A workflow definition must be published to activate it. For a newly created workflow, this will be *false*.
WorkflowType	A string indicating the type of workflow definition. The currently supported types are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. • **Enrollment (Including Renewals)** The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.

Name	Description
	• **Revocation** The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.
Steps	An array of objects indicating the steps in the workflow definition. The contents of each step will vary depending on the type of workflow and the type of step. For a newly created workflow, there will be no data in this value. Possible steps include:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition step.
DisplayName	A string indicating the display name for the step.
UniqueName	A string indicating the unique name for the step. This value must be unique among the steps in the particular workflow definition. It is intended to be used as a user-friendly reference ID.
ExtensionName	A string indicating the type of step. The currently supported types are: • **Email** Send an email message. This is a separate email message from those typically sent as part of a *Require Approval* step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request. • **PowerShell** Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including: • ConvertFrom-Csv • ConvertFrom-Json • ConvertFrom-Markdown

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>• ConvertFrom-SddlString • ConvertFrom-StringData • ConvertTo-Csv • ConvertTo-Html • ConvertTo-Json • ConvertTo-Xml • ForEach-Object • Get-Command • Where-Object • **CustomPowerShell** Run a PowerShell script that has been imported into the Keyfactor Command database. All scripts in the database that have been configured with the workflow category will be available for use. See Extensions Scripts on page 1704 for adding scripts to the database. • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step can require approval from just one approver or multiple approvers. The workflow will be suspended at this point until the correct number of approvals from users with the correct security roles is received or until one deny is received before continuing to the next step. As part of this step, an email message is sent indicating whether the step was approved or denied—typically to the requester. This step does not include logic to send an email initiating the approval process (letting users know something needs approval). Use an *Email* type step for this. **Important:** Workflows are not supported with CA delegation when they contain steps that require approval. For more information, see the CA configuration Authorization Methods Tab on page 367. **Note:** The users that you send email to initiating the approval process must be members of a security role that is allowed to submit signals (approve/deny) for the workflow in order to approve or deny the request.</td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> Tip: The workflow builder does not include a step to send a notification to the requester of a certificate on an enrollment once the certificate is issued by the CA (as opposed to approved in Keyfactor Command). Use the issued alerts for this (see <a>Issued Request Alert Operations on page 189). - RESTRequest Run a REST (API) request using Active Directory as an identity provider and Basic or Windows authentication. The REST request contents are embedded within the step. It does not call out to an external file. - OAuthRESTRequest Run a REST (API) request using an identity provider other than Active Directory and Token authentication. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent (Enrollment Only) On an enrollment (either CSR or PFX), create a resigned CSR to prepare an updated enrollment request for delivery to a Microsoft CA after a previous step in the workflow has been used to update either the SANs in the initial request, subject (DN) in the initial request or both. This step must be placed later in the workflow than the step(s) to modify the SANs and/or subject. The SANs and subject may be modified with either of the PowerShell step types or a custom step type. The step creates a new CSR using the same public key as the original CSR using the updated SAN and/or subject values. It signs the new CSR with the certificate provided in the step's configuration. For this type of step you will need an enrollment agent certificate available as a PKCS#12 (.PFX) file with included private key to import into Keyfactor Command. This can be a user certificate or a computer certificate (e.g. generated from a copy of the Enrollment Agent template or the Enrollment Agent (Computer) template) and must have a Certificate Request Agent EKU. Note that the built-in Enrollment Agent and Enrollment Agent (Computer) templates do not allow private keys to be exported by default. You will need a template that allows private key export or will need to manually override private key export to create a certificate with an exportable private key in order to create a PKCS#12 (.PFX) file. </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td> **Important:** This step applies to Microsoft CAs only. If this step is added to workflow for requests directed to an EJBCA CA, it will fail on enrollment. Note that EJBCA supports submission of updated SAN or subject details as part of standard functionality. - SubjectFormatter (Enrollment Only) On an enrollment done through the Keyfactor Windows Enrollment Gateway using a client-side template configured with the Build from this Active Directory information option on the template, this workflow step handles formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as Build from this Active Directory information must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. Important: The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the Keyfactor Windows Enrollment Gateway Installation and Configuration Guide. - EnrollStep Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. - NOOPStep An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. - RevokeStep Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately </td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>following the EndNOOP step.

Tip: For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 2294) and are not configured individually in the workflow steps.</td></tr>
<tr><td>Enabled</td><td>A Boolean indicating whether the step is enabled to run (true) or not (false).</td></tr>
<tr><td>Config-urationPara-meters</td><td>An object containing the configuration parameters for the workflow definition step. These will vary depending on the type of workflow and the type of step (see ExtensionName).

Note: There are no ConfigurationParameters for steps of type SubjectFormatter, EnrollStep, NOOPStep, or RevokeStep.

Possible CustomPowerShell parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>ScriptParameters</td><td>An object defining any parameters to be used in the PowerShell script.</td></tr>
<tr><td>ScriptName</td><td>A string containing the Keyfactor Command reference name of PowerShell script as stored in the Keyfactor Command database (see Extensions Scripts on page 1704).
A sample PowerShell script (CustomPower-shellExample.ps1) is provided in the \Exten-sionLibrary\net6.0\Workflow directory on the Keyfactor Command server under the install directory. By default, this is:

<code>C:\Program Files\Keyfactor\Keyfactor Platform \ExtensionLibrary\net6.0\Workfl ow</code></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append additional data to it using PowerShell.

Possible **Email** parameters include:

Value	Description
Subject	A string indicating the subject line for the email message that will be delivered when the workflow definition step is executed.
Message	A string indicating the email message that will be delivered when the workflow definition step is executed. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. For example, for an enrollment pending request notification: "Hello,\n\nA certificate using the $(template) template was requested by $(requester:displayname) from $(CA) on $(subdate). The certificate details include:\n\n<table>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: $(request:cn)</td><td>App Owner First Name: $(metadata:ApOwnerFirstName)</td></tr>\n<tr><td>DN: $(request:dn)</td><td>App Owner Last Name: $(metadata:ApOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:ApOwnerEmailAddress)</td></tr>\n<tr><td> </td><td>Business

Name	Description

Name	Description

Value	Description
	Critical: $(metadata:BusinessCritical)</td></tr>\n\nPlease review this request and issue the certificate as appropriate by going here:\n\n$(reviewlink)\n\nThanks!\n\nYour Certificate Management Tool\n” See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. **Note:** The $(requester:displayname) substitutable special text token is only supported in environments using Active Directory as an identity provider.
Recipients	An array of strings containing the recipients for the workflow definition email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from

Name	Description

Name	Description

the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester).

Possible **EnrollmentAgent** parameters include:

Value	Description
EnrollmentAgentCert	A string containing the base-64-encoded representation of the enrollment agent certificate with private key (in PKCS#12 format) that will be used to sign the CSR. This can be either a user certificate or a computer certificate and must have a Certificate Request Agent EKU.
EnrollmentAgentCertPassword	An object indicating the password information used to secure the private key of the enrollment agent certificate. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database.

Name	Description

Name	Description

Value	Description
	• Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.

Possible **PowerShell** parameters include:

Value	Description
ScriptParameters	An object defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>additional data to it using PowerShell. Possible **RequireApproval** parameters include: <table><tr><th>Value</th><th>Description</th></tr><tr><td>MinimumApprovals</td><td>In integer indicating the minimum number of users who must approve the request to allow the request to complete.</td></tr><tr><td>DenialEmailSubject</td><td>A string indicating the subject line for the email message that will be delivered if the request is denied.</td></tr><tr><td>DenialEmailMessage</td><td>A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens.</td></tr><tr><td>DenialEmailRecipients</td><td>An array of strings containing the recipients for the denial email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.</td></tr></table></td></tr></table>

Name	Description
	Name / **Description** **Value** / **Description** **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:ApOwnerEmailAddress). ApprovalEmailSubject — A string indicating the subject line for the email message that will be delivered if the request is approved. ApprovalEmailMessage — A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See Table 11: Tokens for Workflow Definitions for a complete list of available tokens. ApprovalEmailRecipients — An array of strings containing the recipients for the approval email. Each email message can have multiple recipients. You can use specific email addresses and/or use tokens to replace an email address variable with actual email addresses at processing time. Available email tokens include: • $(requester:mail) The certificate requester, based on

Name	Description

Name	Description

Value	Description
	a lookup in Active Directory of the email address associated with the requester on the certificate. **Note:** The $(requester:mail) substitutable special text token is only supported in environments using Active Directory as an identity provider. • Your custom email-based metadata field, which would be specified similarly to $(metadata:ApOwnerEmailAddress).

Tip: Tokens (a.k.a. substitutable special text) may be used in the subject line, message and email recipient fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can select $(requester) in the workflow definition for an enrollment request and the email message will contain the specific certificate requester name instead of the variable $(requester).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header.

Name	Description
	Name / **Description**

Name

Description

Value	Description
	For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBucketProperty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)`

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
UseBasic-Auth	A Boolean indicating whether Basic authentication should be used for the request (True) or not (False). If *UseBasicAuth* is *False*, Windows authentication in the context of the Keyfactor Command application pool user will be used (see Create Service Accounts for Keyfactor Command on page 2757).
BasicUser-name	An object indicating the username information to use for authentication if *UseBasicAuth* is *True*. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned in responses.
BasicPass-word	An object indicating the password information to use for authentication if *UseBasicAuth* is *True*. The syntax is the same as for *BasicUsername*.

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	Due to its sensitive nature, this value is not returned in responses.

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example:
	`192.168.12.0/24,192.168.14.22/24`
	When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.

ContentTy-	A string indicating the content type for the request.

Name	Description

Name	Description

Value	Description
pe	Supported values are: • application/json

RequestC-ontent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):
	
	`{`
	`"Id": "$(certid)",`
	`"Metadata":{`
	`"RevocationComment": "$(cmnt)"`
	`}`
	`}`
	
	Note: This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).

Possible **RestRequest** parameters include:

Value	Description
Headers	An object containing the header information for the request. The key is the name of the specific request header (for Keyfactor API request headers, see

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Table 90: Common Request Headers and the specific documentation for each endpoint) and the value is the value that should be set for that header. For a Keyfactor API request, this might look like: `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBuck-etProp-erty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the

Name	Description
	Name / **Description**

Name

Description

Value	Description
	following: `$(MyResponse.[0].ClientMachine)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE
client_id	A string indicating the ID of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844).
client_secret	An object indicating the secret of the identity provider client that should be used to authenticate the session (see Authenticating to the Keyfactor API on page 844). Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. Due to its sensitive nature, this value is not returned

Name	Description
	Name / **Description**

Value	Description
	in responses.
TokenEndpoint	A string container the URL of the token endpoint for your identity provider. For example: `https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token` For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

URL	A string containing the URL for the request, including tokens, if desired. For a Keyfactor API request, this might look like:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL`
	Or, with tokens:
	`https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)`
	Note: To prevent REST requests from being made to inappropriate locations by malicious users, configure a system environment variable of KEYFACTOR_BLOCKED_

Name	Description

Name	Description

Value	Description
	OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example: `192.168.12.0/24,192.168.14.22/24` When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.
ContentType	A string indicating the content type for the request. Supported values are: • application/json

RequestContent	A string containing the body of the REST request, if needed. For a Keyfactor API request, this will vary depending on the request and might look like (for a PUT /Certificates/Metadata request):
	
	`{`
	`"Id": "$(certid)",`
	`"Metadata":{`
	`"RevocationComment": "$(cmnt)"`
	`}`
	`}`
	
	Note: This example assumes you have a metadata field called RevocationComment.

Tip: Tokens (a.k.a. substitutable special text) may be used in

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the URL and request content fields. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and insert it into a custom metadata field in the certificate by doing a PUT /Certificates/Metadata request for the $(id).</td></tr>
<tr><td>Signals</td><td>An array of objects containing data used at the point in the workflow step where the workflow needs to continue based on user input. These will vary depending on the type of workflow and the type of step (see ExtensionName). Possible RequireApproval values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>RoleIds</td><td>An array of integers indicating the security roles whose members are allowed to approve the request.</td></tr>
<tr><td>SignalName</td><td>A string indicating the name of the signal. This value will vary depending on the workflow step. For the built-in Require Approval step, the SignalName is ApprovalStatus.</td></tr>
</table>
Important: If all the security roles configured for a workflow step are deleted from Keyfactor Command, no users will be able to submit signals for workflow instances initiated with that workflow definition. To remedy this, update the workflow definition with one or more current security roles, re-publish it, and then restart any outstanding workflow instances.</td></tr>
<tr><td>Conditions</td><td>An object containing conditions indicating whether the step should run (true) or not (false). Conditions may either have a static value of True or False or a token that will have a value of True or False at the time the step is run. More than one condition may be added. If multiple conditions are used in the same step, all conditions must have a value of True at the time the step is evaluated to be run in order for the step to run. If any single condition evaluates to False, the step will not run. Condition values are:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the condition.</td></tr>
<tr><td>Value</td><td>A string indicating the value of the condition. This should be one of true, false, or a token that will be set to either true or false in an earlier step in the workflow (see <a>Adding, Copying or Modifying a Workflow Definition on page 237 for an example).</td></tr>
</table>
</td></tr>
<tr><td>Outputs</td><td>An object indicating the next step in the workflow. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>continue</td><td>A string indicating the UniqueName of the next workflow step in the chain. This value will be null for the final step in the chain.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>DraftVersion</td><td>An integer indicating the version number of the workflow definition. If this version number does not match the PublishedVersion, changes have been made to the workflow definition that have not yet been published.</td></tr>
<tr><td>PublishedVersion</td><td>An integer indicating the currently published version number of the workflow definition. For a newly created workflow, this value will be null.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.41 Workflow Instances

The Workflow Instances component of the Keyfactor API includes methods necessary to programmatically retrieve, restart, delete and submit data into workflow instances.

Table 813: Workflow Instances Endpoints

Endpoint	Method	Description	Link
/{instanceId}	DELETE	Delete the workflow instance with he specified GUID.	DELETE Workflow Instances Instance Id below
/{instanceId}	GET	Retrieve the workflow instance with the specified GUID.	GET Workflow Instances Instance ID on the next page
/	GET	Retrieve a list of the workflow instances.	GET Workflow Instances on page 2655
/My	GET	Retrieve the workflow instances created by the user making the API request.	GET Workflow Instances My on page 2660
/AssignedToMe	GET	Retrieve the workflow instances assigned to the user making the API request.	GET Workflow Instances AssignedToMe on page 2665
/{instanceId}/Stop	POST	Rejects a workflow instance, preventing it from continuing.	POST Workflow Instances Instance Id Stop on page 2670
/{instanceId}/Signals	POST	Input data to the workflow instance with the specified GUID.	POST Workflow Instances Instance ID Signals on page 2671
/{instanceId}/Restart	POST	Restart the specified workflow instance after a failure.	POST Workflow Instances Instance Id Restart on page 2673

3.6.41.1 DELETE Workflow Instances Instance Id

The DELETE /Workflow/Instances/{instanceId} method is used to delete the workflow instance with the specified GUID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/instances/manage/

Table 814: DELETE Workflow Instances {instanceid} Input Parameters

Name	In	Description
instanceId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow instance to delete. Use the *GET /Workflow/Instances* method (see GET Workflow Instances on page 2655) to retrieve a list of all the workflow instances to determine the GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.41.2 GET Workflow Instances Instance ID

The GET /Workflow/Instances/{instanceId} method is used to retrieve the initiated workflow with the specified instance GUID. Both in progress and completed workflows will be returned. This method returns HTTP 200 OK on a success with details about the workflow instance.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/instances/read/
OR
/workflows/instances/read/pending/
OR
/workflows/instances/read/mine/

Users with */mine/* or */pending/* will only be able to retrieve the workflow instances created by them (*/mine/*) or assigned to them (*/pending/*) unless they also have the higher level just */read/*.

Table 815: GET Workflow Instances {instanceId} Input Parameters

Name	In	Description
instanceId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow instance to retrieve. Use the *GET /Workflow/Instances* method (see GET Workflow Instances on page 2655) to retrieve a list of all the workflow instances to determine the GUID. Note that the **integer** workflow IDs (returned with *GET /Workflow/Instances/{instanceId}*) cannot be used with the API; only the **GUID** from *GET /Workflow/Instances* can be used in this case.

Table 816: GET Workflow Instances {instanceId} Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow instance.
Status	A string indicating the current status of the workflow instance. The possible statuses are: • CanceledForRestart • Complete • Failed • Rejected • Running • Suspended
CurrentStepID	A string indicating the Keyfactor Command reference GUID of the workflow instance step.
StatusMessage	A string indicating the current status message for the workflow instance. Possible status messages vary and may include: • Access is denied • Awaiting # more approval(s) from approval roles. • Either the credentials are invalid, or the CA on [*CA hostname*] is not running • Issued • Issued. The private key was successfully retained. • Post-process Failed: *[Message indicating reason for failure generally from the CA*] • Pre-process failed: *[Message indicating details of the failure*] • Revoked • Step 'Keyfactor-Enroll' failed: *[Message indicating details of the failure*] • Step 'Keyfactor-Revoke' failed:*[Message indicating details of the failure*] • Step [*custom step name*] failed: [*Message indicating details of the failure*] • Taken Under Submission. The certificate template requires manager approval, and is marked as pending. • Workflow rejected by user with Id #.
Signals	An array of objects containing the data used at the point in the workflow step where the workflow needs to continue based on user input. These will vary depending on the type of workflow and the type of step. Possible **RequireApproval** values are: **Value** / **Description** SignalName — A string indicating the name of the signal. For a RequireApproval step, this is *ApprovalStatus*.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>StepSignalId</td><td>A string indicating the Keyfactor Command reference GUID of the signal in the step.</td></tr>
<tr><td>SignalReceived</td><td>A Boolean indicating whether a signal (input) has been received from at least one end user (true) or not (false).
For a RequireApproval workflow that requires approval from more than one user, the SignalReceived may be true while the workflow instance still has a Suspended status indicating further input is needed.</td></tr>
</table>
</td></tr>
<tr><td>Definition</td><td>An object containing the workflow definition. Workflow definition data includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the workflow definition.</td></tr>
<tr><td>DisplayName</td><td>A string indicating the display name defined for the workflow definition.</td></tr>
<tr><td>Version</td><td>An integer indicating the version number of the workflow definition.</td></tr>
<tr><td>WorkflowType</td><td>A string indicating the type of workflow definition. The currently supported types are:
• CertificateEnteredCollection
The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection.
For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered.
• CertificateLeftCollection
The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate.
• Enrollment (Including Renewals)
The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.
• Revocation
The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.</td></tr>
</table>
</td></tr>
<tr><td>CurrentStepDisplayName</td><td>A string indicating the display name defined for the workflow instance step.</td></tr>
<tr><td>CurrentStepUniqueName</td><td>A string indicating the unique name defined for the workflow instance step. This value is unique among the steps in a particular workflow definition. It is intended to be used as a user-friendly reference ID.</td></tr>
<tr><td>Title</td><td>A string indicating a description for the action taking place in the step, made up of the InitiatingUserName (either DOMAIN\\username or Timer Service) followed by an indication of the type of action and a specific message about the action. For example:
<pre>"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=appsrvr14.keyexample.com."</pre>Or
<pre>"KEYEXAMPLE\\jsmith is revoking certificate with CN=appsrvr12.keyexample.com."</pre></td></tr>
<tr><td>LastModified</td><td>A string indicating the date and time on which the initiated instance was last updated. The instance is updated each time a step in the workflow is completed, when signals are received for a step that accepts signals (e.g. a requires approval step), or when an instance is stopped or restarted.</td></tr>
</table>

Name	Description
StartDate	A string indicating the date and time when the instance was initiated.
InitialData	An object containing the data included in the workflow instance when the workflow was initiated. Initial workflow instance data includes:

Name	Operation Type	Description
CertificateAuthority	Enrollment and Revocation	A string indicating the certificate authority that will be used to enroll against, for enrollment requests, or that issued the certificate, for revocation requests in *hostname\logical name* format.
CertificateId	Certificate Collection and Revocation	An integer indicating the Keyfactor Command reference ID for the certificate.
SerialNumberString	Revocation	A string indicating the serial number of the certificate being revoked.
Thumbprint	Revocation	A string indicating the thumbprint of the certificate being revoked.
RevokeCode	Revocation	An integer containing the specific reason that the certificate is being revoked. Available values are:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>-1</td><td>Remove from Hold</td></tr>
<tr><td>0</td><td>Unspecified</td></tr>
<tr><td>1</td><td>Key Compromised</td></tr>
<tr><td>2</td><td>CA Compromised</td></tr>
<tr><td>3</td><td>Affiliation Changed</td></tr>
<tr><td>4</td><td>Superseded</td></tr>
<tr><td>5</td><td>Cessation of Operation</td></tr>
<tr><td>6</td><td>Certificate Hold</td></tr>
<tr><td>7</td><td>Remove from CRL. Only valid in the case that a cert is already on a CRL in a manner that it can be removed, such as Certificate Hold</td></tr>
</table>

The default is Unspecified.
</td></tr>
<tr><td>EffectiveDate</td><td>Revocation</td><td>A string containing the date and time when the certificate will be revoked.</td></tr>
<tr><td>Comment</td><td>Revocation</td><td>A string containing a freeform reason or comment on why the certificate is being revoked.</td></tr>
<tr><td>Delegate</td><td>Revocation</td><td>A Boolean indicating whether delegation is enabled for the certificate authority that issued the certificate (true) or not (false).</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td>OperationStart</td><td>Revocation</td><td>A string indicating the time at which the revocation workflow was initiated.</td></tr>
<tr><td>Template</td><td>Enrollment</td><td>A string indicating the certificate template short name used for the enrollment request.</td></tr>
<tr><td>IncludeChain</td><td>Enrollment</td><td>A Boolean indicating whether to include the certificate chain in the enrollment response (true) or not (false).</td></tr>
<tr><td>SANs</td><td>Enrollment</td><td>An object indicating the subject alternative names (SANs) for the certificate requested in the enrollment, each type an array of strings. Possible values for the key are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>rfc822</td><td>RFC 822 Name</td></tr>
<tr><td>dns</td><td>DNS Name</td></tr>
<tr><td>directory</td><td>Directory Name</td></tr>
<tr><td>uri</td><td>Uniform Resource Identifier</td></tr>
<tr><td>ip4</td><td>IP v4 Address</td></tr>
<tr><td>ip6</td><td>IP v6 Address</td></tr>
<tr><td>registeredid</td><td>Registered ID (an OID)</td></tr>
<tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr>
<tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr>
</table>
For example:</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Operation Type	Description

		<pre>"SANs": {
		  "dns": [
		  "dnssan1.keyexample.com",
		  "dnssan2.keyexample.com",
		  "dnssan3.keyexample.com"
		  ],
		  "ip4": [
		  "192.168.2.73"
		  ]
		}</pre>

AdditionalAttributes	Enrollment	An object indicating values for any custom enrollment fields set on the certificate template to supply custom request attributes to the CA during the enrollment process.
Metadata	Enrollment	An object indicating values for the metadata fields that will be associated with the certificate once it is in Keyfactor Command. The *key* is the field name and the *value* is the value for the field.
Format	Enrollment	A string indicating the desired output format for the certificate. A value of STORE indicates that the certificate is intended to be delivered into one or more certificate stores.
CustomName	Enrollment	A string indicating a custom friendly name for the certificate.
Subject	Enrollment	A string containing the subject name of the requested certificate using X.500 format.
RenewalCertificate	Enrollment	An object containing the certificate information for the certificate that is being renewed. Certificate data includes:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Certificate</td><td>An object referencing the certificate being renewed in the following format:

<pre>{
"RawData":"[PEM-encoded certificate string]"
}</pre></td></tr>
<tr><td>CertificateId</td><td>An integer containing the Keyfactor Command reference ID of the certificate being renewed.</td></tr>
</table>

Note: This field is only populated for enrollments that are generated by requesting a certificate renewal (see <a>Renew on page 69 in the Keyfactor Command Reference Guide and <a>POST Enrollment Renew on page 1694).
</td></tr>
<tr><td>Stores</td><td>Enrollment</td><td>An array of objects indicating the certificate stores to which the certificate should be distributed. Store details include:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreId</td><td>A string indicating the certificate store(s) to which the certificate should be deployed.

Use the GET /CertificateStores method (see</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Operation Type** / **Description** **Name** / **Description** GET Certificate Stores on page 1327) with a query of "Approved -eq true" to retrieve a list of all your approved certificate stores to determine the GUID(s) of the store(s). Alias: A string indicating the alias of the certificate upon entry into the store. The format of and requirement for this varies depending on the certificate store type and whether the *Overwrite* flag is selected. See PFX Enrollment on page 146 in the *Keyfactor Command Reference Guide* for more information. Overwrite: A Boolean that sets whether a certificate in the store with the *Alias* provided should be overwritten with the new certificate (true) or not (false). The default is *false*. Use the *GET /Certificates/Locations/{id}* method (see GET Certificates Locations ID on page 1116) to retrieve a list of the locations an existing certificate is in to determine the alias used for the certificate in the certificate store. Properties: An object containing the unique entry parameters defined for the certificate store type that need to be populated for the certificate. The *key* is the name of the specific entry parameter from the certificate store type definition as returned in the JobProperties on the store type using the *GET Certi-*

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>ficateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for CitrixAdc, the key name that is optionally used to associate the certificate with a virtual server is virtualServerName and is returned by GET CertificateStoreTypes like so:
<pre>"JobProperties": ["sniCert", "virtualServerName"]</pre>It can be seen in the Keyfactor Command Management Portal when editing the certificate store type on the Entry Parameters tab.
The setting is referenced using the following format:
<pre>"Properties": {"sniCert":"MyCertificateName", "virtualServerName":"MyVirtualServerName"}</pre></td></tr>
</table>
</td></tr>
<tr><td>ManagementJobTime</td><td>Enrollment</td><td>An object indicating the schedule for the management job to add the certificate to the certificate store(s). Possible management job time values include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job sched-</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>uled to run immediately (true) or not (false).

Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
<tr><td>ExactlyOnce</td><td>A dictionary that indicates a job scheduled to run at the time specified with the parameter:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).</td></tr>
</table>

For example, exactly once at 11:45 am:

<pre>"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>}

Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>IsPFX</td><td>Enrollment</td><td>A Boolean indicating whether the certificate enrollment type that initiated the workflow instance was PFX (true) or CSR (false).</td></tr>
<tr><td>PfxPasswordSecretInstanceId</td><td>Enrollment</td><td>A string indicating the Keyfactor Command reference GUID for the PFX password used to secure the PFX file on download.</td></tr>
<tr><td>InitiatingUserName</td><td>Certificate Collection, Enrollment and Revocation</td><td>A string indicating the name of the user who initiated the workflow, generally in DOMAIN\\username format.</td></tr>
</table>
</td></tr>
<tr><td>CurrentStateData</td><td>An object containing the data included in the workflow instance as it progresses. This will include data input from PowerShell scripts, REST requests, and signals along with the initial data. Current state workflow instance data includes:</td></tr>
</table>

Name	Description

Name	Operation Type	Description
CertificateAuthority	Enrollment and Revocation	A string indicating the certificate authority that will be used to enroll against, for enrollment requests, or that issued the certificate, for revocation requests.
CertificateId	Certificate Collection and Revocation	An integer indicating the Keyfactor Command reference ID for the certificate.
SerialNumberString	Revocation	A string indicating the serial number of the certificate being revoked.
Thumbprint	Revocation	A string indicating the thumbprint of the certificate being revoked.
RevokeCode	Revocation	An integer containing the specific reason that the certificate is being revoked. Available values are:

Value	Description
-1	Remove from Hold
0	Unspecified
1	Key Compromised
2	CA Compromised
3	Affiliation Changed
4	Superseded
5	Cessation of Operation

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>6</td><td>Certificate Hold</td></tr>
<tr><td>7</td><td>Remove from CRL. Only valid in the case that a cert is already on a CRL in a manner that it can be removed, such as Certificate Hold</td></tr>
</table>

The default is Unspecified.</td></tr>
<tr><td>EffectiveDate</td><td>Revocation</td><td>A string containing the date and time when the certificate will be revoked.</td></tr>
<tr><td>Comment</td><td>Revocation</td><td>A string containing a freeform reason or comment on why the certificate is being revoked.</td></tr>
<tr><td>Delegate</td><td>Revocation</td><td>A Boolean indicating whether delegation is enabled for the certificate authority that issued the certificate (true) or not (false).</td></tr>
<tr><td>OperationStart</td><td>Revocation</td><td>A string indicating the time at which the revocation workflow was initiated.</td></tr>
<tr><td>Template</td><td>Enrollment</td><td>A string indicating the short certificate template name used for the enrollment request.</td></tr>
<tr><td>IncludeChain</td><td>Enrollment</td><td>A Boolean that indicates whether to include the certificate chain in the enrollment response (true) or not (false).</td></tr>
<tr><td>SANs</td><td>Enrollment</td><td>An object indicating the subject alternative names (SANs) for the certificate requested in the enrollment, each type an array of strings. Possible values for the key are:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>rfc822</td><td>RFC 822 Name</td></tr>
<tr><td>dns</td><td>DNS Name</td></tr>
<tr><td>directory</td><td>Directory Name</td></tr>
<tr><td>uri</td><td>Uniform Resource Identifier</td></tr>
<tr><td>ip4</td><td>IP v4 Address</td></tr>
<tr><td>ip6</td><td>IP v6 Address</td></tr>
<tr><td>registeredid</td><td>Registered ID (an OID)</td></tr>
<tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr>
<tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr>
</table>
For example:
<pre>
"SANs": {
   "dns": [
      "dnssan1.keyexample.com",
      "dnssan2.keyexample.com",
      "dnssan3.keyexample.com"
   ],
   "ip4": [
      "192.168.2.73"
   ]
}
</pre>
</td></tr>
<tr><td>AdditionalAttributes</td><td>Enrollment</td><td>An object indicating values for any custom enrollment fields set on the certificate template to supply custom request attributes to the CA during the enrollment process.</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Operation Type	Description
Metadata	Enrollment	An object indicating values for the metadata fields that will be associated with the certificate once it is in Keyfactor Command. The *key* is the field name and the *value* is the value for the field.
Format	Enrollment	A string indicating the desired output format for the certificate. A value of STORE indicates that the certificate is intended to be delivered into one or more certificate stores.
CustomName	Enrollment	A string indicating a custom friendly name for the certificate.
Subject	Enrollment	A string containing the subject name of the requested certificate using X.500 format.
RenewalCertificate	Enrollment	An object containing the certificate information for the certificate that is being renewed. Certificate data includes:

Name	Description
Certificate	An object referencing the certificate being renewed in the following format: `{ "RawData":"[PEM-encoded certificate string]" }`
CertificateId	An integer containing the Keyfactor Command reference ID of the certificate being renewed.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Oper-ation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>Note: This field is only populated for enrollments that are generated by requesting a certificate renewal (see <a>Renew on page 69 in the Keyfactor Command Reference Guide and <a>POST Enrollment Renew on page 1694).</td></tr>
<tr><td>Stores</td><td>Enroll-ment</td><td>An array of objects indicating the certificate stores to which the certificate should be distributed. Store details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreId</td><td>A string indicating the certificate store(s) to which the certificate should be deployed.
Use the GET /CertificateStores method (see <a>GET Certificate Stores on page 1327) with a query of “Approved -eq true” to retrieve a list of all your approved certificate stores to determine the GUID(s) of the store(s).</td></tr>
<tr><td>Alias</td><td>A string indicating the alias of the certificate upon entry into the store. The format of and requirement for this varies depending on the certificate store type and whether the Overwrite flag is selected. See <a>PFX Enrollment on page 146 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Over-</td><td>A Boolean that sets whether a certi-</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

	<table><tr><th>Name</th><th>Operation Type</th><th>Description</th></tr><tr><td></td><td></td><td><table><tr><th>Name</th><th>Description</th></tr><tr><td>write</td><td>ficate in the store with the Alias provided should be overwritten with the new certificate (true) or not (false). The default is false.
	Use the GET /Certificates/Locations/{id} method (see GET Certificates Locations ID on page 1116) to retrieve a list of the locations an existing certificate is in to determine the alias used for the certificate in the certificate store.</td></tr><tr><td>Properties</td><td>An object for the unique parameters defined for the certificate store type that need to be populated for the certificate. The key is the name of the specific parameter from the certificate store type definition as returned in the JobProperties on the store type using the GET CertificateStoreTypes method and the value is the value that should be set for that parameter on the certificate in the certificate store. For example, for NetScaler, the key name that is optionally used to associate the certificate with a virtual server is NetscalerVserver and is returned by GET CertificateStoreTypes like so:
	<pre>"JobProperties": [
	"NetscalerVserver"]</pre></td></tr></table></td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Oper-ation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>It can be seen in the Keyfactor Command Management Portal when editing the certificate store type in the field for Management Job Custom Fields.
The setting is referenced using the following format:
<pre>"Properties": {"NetscalerVserver":"My VirtualServerName"}</pre>
Note: The only built-in certificate store type that makes use of properties that can be set on a certificate-by-certificate basis in the store is NetScaler. You may have custom certificate store types that make use of this functionality.</td></tr>
</table>
</td></tr>
<tr><td>ManagementJobTime</td><td>Enrollment</td><td>An object indicating the schedule for the management job to add the certificate to any certificate store(s). Possible management job time values include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Operation Type	Description

Name	Description
	Tip: In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
ExactlyOnce	A dictionary that indicates a job scheduled to run at the time specified with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2023-11-19T16:23:01Z).

For example, exactly once at 11:45 am:

```
"ExactlyOnce": {
   "Time": "2023-11-27T11:45:00Z"
}
```

Name	Description

Name	Operation Type	Description
		Name: Description: **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *ExactlyOnce*.
IsPFX	Enrollment	A Boolean indicating whether the certificate enrollment type that initiated the workflow instance was PFX (true) or CSR (false).
PfxPasswordSecretInstanceId	Enrollment	A string indicating the Keyfactor Command reference GUID for the PFX password used to secure the PFX file on download.
InitiatingUserName	Certificate Collection, Enrollment and Revocation	A string indicating the name of the user who initiated the workflow, generally in DOMAIN\\username format.
KeyRetention	Enrollment	A Boolean indicating whether the private key for the certificate resulting from the enrollment will be retained in Keyfactor Command (true) or not (false).
CSR	Enrollment	A string containing the CSR generated for the certificate request.
(Custom)	Enrollment and	Optional user-generated custom fields returning response data from PowerShell scripts or REST requests.

Name	Description

Name	Operation Type	Description
	Revocation	
CACertificate	Enrollment	An object containing the certificate information returned from the CA for the certificate that is being requested. CA certificate details include:

Name	Description
CACertificateId	A string containing the ID assigned to the certificate by the CA.
CARequestID	A string containing the ID assigned to the certificate request by the CA.
Status	An integer indicating the status for the certificate as returned by the CA.
Certificate	A string containing the certificate as returned by the CA in base-64 encoded binary format.
CertificateTemplate	A string indicating the certificate template used to issue the certificate.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Oper-ation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>RevocationDate</td><td>A string indicating the revocation date for the certificate as returned by the CA.</td></tr>
<tr><td>RevocationReason</td><td>A string indicating the revocation reason for the certificate as returned by the CA.</td></tr>
<tr><td>ArchivedKey</td><td>A Boolean indicating whether the certificate is configured for key archival on the CA (true) or not (false).</td></tr>
</table>
Note: This field is only populated only after the certificate has been issued by the CA.
</td></tr>
<tr><td>DispositionMessage</td><td>Enrollment</td><td>A string indicating a message about the certificate request (e.g. “The private key was successfully retained.”).
Note: This field is only populated only after the certificate request has been submitted to the CA.</td></tr>
<tr><td>CACertificateRequest</td><td>Enrollment</td><td>An object containing the certificate information for the certificate that is being requested. Certificate request data includes:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Oper-ation Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CARequestId</td><td>A string containing the ID assigned to the certificate request by the CA.</td></tr>
<tr><td>CSR</td><td>A string containing the certificate signing request for the certificate request as returned by the CA.</td></tr>
<tr><td>Status</td><td>An integer indicating the status for the certificate as returned by the CA.</td></tr>
<tr><td>RequesterName</td><td>A string containing the requester name on the certificate request as returned by the CA.</td></tr>
</table>
Note: This field is populated only if the certificate request fails at the CA level or requires manager approval at the CA level.
</td></tr>
<tr><td>SerialNumber</td><td>Enroll-ment</td><td>A string indicating the serial number of the certificate.</td></tr>
<tr><td>IssuerDn</td><td>Enroll-ment</td><td>A string indicating the distinguished name of the issuer.</td></tr>
<tr><td>Thumbprint</td><td>Enroll-ment</td><td>A string indicating the thumbprint of the certificate.</td></tr>
<tr><td>KeyfactorId</td><td>Enroll-ment</td><td>An integer indicating the Keyfactor Command reference ID for the certificate.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td>KeyStatus</td><td>Enrollment</td><td>An integer indicating the status of the private key retention for the certificate within Keyfactor Command. Possible values are:
0—Unknown
1—Saved
2—Expected
3—NoRetention
4—Failure
5—Temporary</td></tr>
<tr><td>PrivateKeyConverter</td><td>Enrollment</td><td>An internally used Keyfactor Command field.</td></tr>
</table>
</td></tr>
<tr><td>ReferenceId</td><td>A integer indicating the Keyfactor Command reference ID for the workflow instance.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.41.3 GET Workflow Instances

The GET /Workflow/Instances method is used to retrieve the list of workflows that have been initiated. Both in progress and completed workflows are included. This method returns HTTP 200 OK on a success with details about the workflow instances.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/instances/read/

Table 817: GET Workflow Instances Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Workflow Instances Search Feature on page 321. The query fields supported for this endpoint are: • DefinitionId (workflow definition ID) • Id (workflow instance GUID) • InitiatingUserName (DOMAIN\\username) • LastModified • ReferenceId (workflow instance integer ID) • StartDate • Status • Title • WorkflowType
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *CurrentStepDisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 818: GET Workflow Instances Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow instance.
Status	A string indicating the current status of the workflow instance. The possible statuses are: • CanceledForRestart • Complete • Failed • Rejected • Running • Suspended
CurrentStepID	A string indicating the Keyfactor Command reference GUID of the workflow instance step.
StatusMessage	A string indicating the current status message for the workflow instance. Possible status messages vary and may include: • Access is denied • Awaiting # more approval(s) from approval roles. • Either the credentials are invalid, or the CA on [*CA hostname*] is not running • Issued • Issued. The private key was successfully retained. • Post-process Failed: *[Message indicating reason for failure generally from the CA*] • Pre-process failed: *[Message indicating details of the failure*] • Revoked • Step 'Keyfactor-Enroll' failed: *[Message indicating details of the failure*] • Step 'Keyfactor-Revoke' failed:*[Message indicating details of the failure*] • Step [*custom step name*] failed: [*Message indicating details of the failure*] • Taken Under Submission. The certificate template requires manager approval, and is marked as pending. • Workflow rejected by user with Id #.
Definition	An object containing the workflow definition. Workflow definition data includes:

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the workflow definition.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name defined for the workflow definition.</td></tr><tr><td>Version</td><td>An integer indicating the version number of the workflow definition.</td></tr><tr><td>WorkflowType</td><td>A string indicating the type of workflow definition. The currently supported types are: - CertificateEnteredCollection The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. - CertificateLeftCollection The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. - Enrollment (Including Renewals) The workflow is initiated by enrollment for a new or renewed certificate. Steps during </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request. • **Revocation** The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.</td></tr></table>
CurrentStepDisplayName	A string indicating the display name defined for the workflow instance step.
CurrentStepUniqueName	A string indicating the unique name defined for the workflow instance step. This value is unique among the steps in a particular workflow definition. It is intended to be used as a user-friendly reference ID.

Title	A string indicating a description for the action taking place in the step, made up of the *InitiatingUserName* (either DOMAIN\\username or Timer Service) followed by an indication of the type of action and a specific message about the action. For example:
	`"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=appsrvr14.keyexample.com."`
	Or
	`"KEYEXAMPLE\\jsmith is revoking certificate with CN=appsrvr12.keyexample.com."`

LastModified	A string indicating the date and time on which the initiated instance was last updated. The instance is updated each time a step in the workflow is completed, when signals are received for a step that accepts signals (e.g. a requires approval step), or when an instance is stopped or restarted.
StartDate	A string indicating the date and time when the instance was initiated.
ReferenceId	A integer indicating the Keyfactor Command reference ID for the workflow instance.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.41.4 GET Workflow Instances My

The GET /Workflow/Instances/My method is used to retrieve the list of initiated workflows created by the user making the API request—as a result of enrolling for a certificate, for example, or revoking a certificate. This method returns HTTP 200 OK on a success with details about the workflow instances.

Note: If a workflow instance is initiated for a workflow definition that has more than one step requiring input (signals), a user can only provide that input (e.g. approve or deny a require approval request) at the step in the workflow instance where the workflow instance was suspended pending input. The user cannot jump ahead and provide input for future steps in the workflow that have not yet occurred.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/instances/read/
OR
/workflows/instances/read/mine/

Table 819: GET Workflow Instances My Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Workflow Instances Search Feature on page 321. The query fields supported for this endpoint are: • DefinitionId (workflow definition ID) • Id (workflow instance GUID) • InitiatingUserName (DOMAIN\\username) • LastModified • ReferenceId (workflow instance integer ID) • StartDate • Status • Title • WorkflowType
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *Id*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 820: GET Workflow Instances My Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow instance.
Status	A string indicating the current status of the workflow instance. The possible statuses are: • CanceledForRestart • Complete • Failed • Rejected • Running • Suspended
CurrentStepID	A string indicating the Keyfactor Command reference GUID of the workflow instance step.
StatusMessage	A string indicating the current status message for the workflow instance. Possible status messages vary and may include: • Access is denied • Awaiting # more approval(s) from approval roles. • Either the credentials are invalid, or the CA on [*CA hostname*] is not running • Issued • Issued. The private key was successfully retained. • Post-process Failed: *[Message indicating reason for failure generally from the CA*] • Pre-process failed: *[Message indicating details of the failure*] • Revoked • Step 'Keyfactor-Enroll' failed: *[Message indicating details of the failure*] • Step 'Keyfactor-Revoke' failed:*[Message indicating details of the failure*] • Step [*custom step name*] failed: [*Message indicating details of the failure*] • Taken Under Submission. The certificate template requires manager approval, and is marked as pending. • Workflow rejected by user with Id #.
Definition	An object containing the workflow definition. Workflow definition data includes:

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID of the workflow definition.</td></tr><tr><td>DisplayName</td><td>A string indicating the display name defined for the workflow definition.</td></tr><tr><td>Version</td><td>An integer indicating the version number of the workflow definition.</td></tr><tr><td>WorkflowType</td><td>A string indicating the type of workflow definition. The currently supported types are: - CertificateEnteredCollection The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. - CertificateLeftCollection The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the removal of a web server certificate. - Enrollment (Including Renewals) The workflow is initiated by enrollment for a new or renewed certificate. Steps during </td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.
• Revocation
The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.</td></tr>
</table>
</td></tr>
<tr><td>CurrentStepDisplayName</td><td>A string indicating the display name defined for the workflow instance step.</td></tr>
<tr><td>CurrentStepUniqueName</td><td>A string indicating the unique name defined for the workflow instance step. This value is unique among the steps in a particular workflow definition. It is intended to be used as a user-friendly reference ID.</td></tr>
<tr><td>Title</td><td>A string indicating a description for the action taking place in the step, made up of the InitiatingUserName (either DOMAIN\\username or Timer Service) followed by an indication of the type of action and a specific message about the action. For example:
<pre>"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=appsrvr14.keyexample.com."</pre>Or
<pre>"KEYEXAMPLE\\jsmith is revoking certificate with CN=appsrvr12.keyexample.com."</pre></td></tr>
<tr><td>LastModified</td><td>A string indicating the date and time on which the initiated instance was last updated. The instance is updated each time a step in the workflow is completed, when signals are received for a step that accepts signals (e.g. a requires approval step), or when an instance is stopped or restarted.</td></tr>
<tr><td>StartDate</td><td>A string indicating the date and time when the instance was initiated.</td></tr>
<tr><td>ReferenceId</td><td>A integer indicating the Keyfactor Command reference ID for the workflow instance.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.41.5 GET Workflow Instances AssignedToMe

The GET /Workflow/Instances/AssignedToMe method is used to retrieve the list of initiated workflows awaiting input from the user making the API request. This method returns HTTP 200 OK on a success with details about the workflow instances.

Note: If a workflow instance is initiated for a workflow definition that has more than one step requiring input (signals), a user can only provide that input (e.g. approve or deny a require approval request) at the step in the workflow instance where the workflow instance was suspended pending input. The user cannot jump ahead and provide input for future steps in the workflow that have not yet occurred.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/instances/read/
OR
/workflows/instances/read/pending/

Table 821: GET Workflow Instances AssignedToMe Input Parameters

Name	In	Description
QueryString	Query	A string containing a query to limit the results (e.g. field1 -eq value1 AND field2 -gt value2). The default is to return all records. Fields available for querying through the API for the most part match those that appear in the Keyfactor Command Management Portal search drop-downs for the same feature. For querying guidelines, refer to: Using the Workflow Instances Search Feature on page 321. The query fields supported for this endpoint are: • DefinitionId (workflow definition ID) • Id (workflow instance GUID) • InitiatingUserName (DOMAIN\\username) • LastModified • ReferenceId (workflow instance integer ID) • StartDate • Status • Title • WorkflowType
PageReturned	Query	An integer that specifies how many multiples of the returnLimit to skip and offset by before returning results, to enable paging. The default is 1.
ReturnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
SortField	Query	A string containing the property by which the results should be sorted. Fields available for sorting through the API for the most part match those that appear as sortable columns in the Keyfactor Command Management Portal. The default sort field is *CurrentStepDisplayName*.
SortAscending	Query	An integer that sets the sort order on the returned results. A value of 0 sorts results in ascending order while a value of 1 sorts results in descending order. The default is ascending.

Table 822: GET Workflow Instances AssignedToMe Response Data

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow instance.
Status	A string indicating the current status of the workflow instance. The possible statuses are: • CanceledForRestart • Complete • Failed • Rejected • Running • Suspended Only instances with a Status of *Suspended* are returned using this method.
CurrentStepID	A string indicating the Keyfactor Command reference GUID of the workflow instance step.
StatusMessage	A string indicating the current status message for the workflow instance. Possible status messages vary and may include: • Access is denied • Awaiting # more approval(s) from approval roles. • Either the credentials are invalid, or the CA on [*CA hostname*] is not running • Issued • Issued. The private key was successfully retained. • Post-process Failed: *[Message indicating reason for failure generally from the CA*] • Pre-process failed: *[Message indicating details of the failure*] • Revoked • Step 'Keyfactor-Enroll' failed: *[Message indicating details of the failure*] • Step 'Keyfactor-Revoke' failed:*[Message indicating details of the failure*] • Step [*custom step name*] failed: [*Message indicating details of the failure*] • Taken Under Submission. The certificate template requires manager approval, and is marked as pending. • Workflow rejected by user with Id *#*. Only instances with a *StatusMessage* of *Awaiting # more approval(s) from*

Name	Description
	approval roles. are returned using this method.
Definition	An object containing the workflow definition. Workflow definition data includes:

Name	Description
Id	A string indicating the Keyfactor Command reference GUID of the workflow definition.
DisplayName	A string indicating the display name defined for the workflow definition.
Version	An integer indicating the version number of the workflow definition.
WorkflowType	A string indicating the type of workflow definition. The currently supported types are: • **CertificateEnteredCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that now meet the query criteria of the specified certificate collection. For example, when a certificate discovered on an SSL scan becomes part of the Weak Keys collection, an email message can be generated notifying the PKI administrators that a new certificate with a weak key has been discovered. • **CertificateLeftCollection** The workflow is initiated by an automated task that Keyfactor Command runs periodically against your collections to identify certificates that no longer meet the query criteria of the specified certificate collection. For example, when a certificate in the Web Server Certificates collection disappears, a REST request can be made to open a support ticket request to investigate the

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>removal of a web server certificate.
• Enrollment (Including Renewals)
The workflow is initiated by enrollment for a new or renewed certificate. Steps during the workflow can be used to do things such as require manager approval for the enrollment or manipulate the subject and/or SANs for the certificate request.
• Revocation
The workflow is initiated by revoking a certificate. Steps during the workflow can be configured to do things such as modify the revocation comment entered when the certificate is revoked, append an additional comment, and store the resulting extended comment in a metadata field.</td></tr>
</table>
</td></tr>
<tr><td>CurrentStepDisplayName</td><td>A string indicating the display name defined for the workflow instance step.</td></tr>
<tr><td>CurrentStepUniqueName</td><td>A string indicating the unique name defined for the workflow instance step. This value is unique among the steps in a particular workflow definition. It is intended to be used as a user-friendly reference ID.</td></tr>
<tr><td>Title</td><td>A string indicating a description for the action taking place in the step, made up of the InitiatingUserName (either DOMAIN\\username or Timer Service) followed by an indication of the type of action and a specific message about the action. For example:
<pre>"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=appsrvr14.keyexample.com."</pre>Or
<pre>"KEYEXAMPLE\\jsmith is revoking certificate with CN=appsrvr12.keyexample.com."</pre></td></tr>
<tr><td>LastModified</td><td>A string indicating the date and time on which the initiated instance was last updated. The instance is updated each time a step in the workflow is completed, when signals are received for a step that accepts signals (e.g. a requires approval step), or when an instance is stopped or restarted.</td></tr>
</table>

Name	Description
StartDate	A string indicating the date and time when the instance was initiated.
ReferenceId	A integer indicating the Keyfactor Command reference ID for the workflow instance.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.41.6 POST Workflow Instances Instance Id Stop

The POST /Workflow/Instances/{instanceId}/Stop method is used to stop the workflow instance with the specified GUID, preventing it from continuing. This endpoint returns 204 with no content upon success.

Note: Only workflow instances with a Status of *Suspended* can be stopped.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/instances/manage/

Table 823: POST Workflow Instances {instanceid} Stop Input Parameters

Name	In	Description
instanceId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow instance to stop. Use the *GET /Workflow/Instances* method (see GET Workflow Instances on page 2655) to retrieve a list of all the workflow instances to determine the GUID.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.41.7 POST Workflow Instances Instance ID Signals

The POST /Workflow/Instances/{instanceId}/Signals method is used to input signals to the workflow instance with the specified GUID. This endpoint returns 204 with no content upon success.

Note: If a workflow instance is initiated for a workflow definition that has more than one step requiring input (signals), a user can only provide that input (e.g. approve or deny a require approval request) at the step in the workflow instance where the workflow instance was suspended pending input. The user cannot jump ahead and provide input for future steps in the workflow that have not yet occurred.

Note: A locking conflict may occur if two (or more) users attempt to provide input to a workflow instance (e.g. approve a request) at exactly the same time. If this happens, input from only one of the users will be reflected in the Management Portal, and the workflow instance will not be moved along to the next step if it should have been with input from the two users. The other input is still accepted, however, and there is a scheduled task that runs daily and attempts to continue all suspended workflows that may be eligible to continue but have not done so due to locking conflicts.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
The user executing the request must hold at least one security role ID configured in the workflow definition step for which signal data is being input.

Table 824: POST Workflow Instances {instanceid} Signals Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>instanceId</td><td>Path</td><td>Required. A string indicating the Keyfactor Command reference GUID of the workflow instance to which to input a signal.
Use the GET /Workflow/Instances method (see <a>GET Workflow Instances on page 2655) to retrieve a list of all the workflow instances to determine the GUID.</td></tr>
<tr><td>SignalKey</td><td>Body</td><td>Required. A string indicating the key for the signal. This is made up of the unique name for the step within the definition plus the signal type, separated by a period (UniqueName.SignalType). For a Require Approval step, the key input type will be ApprovalStatus, so the full SignalKey will look something like:
<pre>RequireApproval1.ApprovalStatus</pre>Use the GET /Workflow/Definitions/{definitionid} method (see <a>GET Workflow Definitions Definition ID on page 2494) to return workflow details including the workflow steps to determine the UniqueName of the step for which you want to input a signal or one of the GET methods for workflow instances (see <a>GET Workflow Instances on page 2655, <a>GET Workflow Instances AssignedToMe on page 2665, or <a>GET Workflow Instances My on page 2660) to return the CurrentStepUniqueName.</td></tr>
<tr><td>Data</td><td>Body</td><td>Required. An object providing the input information for the signal. The key(s) will vary depending on the signal. RequireApproval signal data values are:
<table><tr><th>Key</th><th>Value</th></tr><tr><td>Approved</td><td>Required. A Boolean indicating whether the request is approved (true) or denied (false).</td></tr><tr><td>Comment</td><td>A string containing a comment to associate with the signal. The maximum comment length is 500 characters.</td></tr></table>For example, to approve a Require Approval step called RequireApproval1 with a comment:
<pre>{
   "SignalKey": "RequireApproval1.ApprovalStatus",
   "Data": {
      "Approved": "True",
      "Comment": "Here is my comment."
   }
}</pre></td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.6.41.8 POST Workflow Instances Instance Id Restart

The POST /Workflow/Instances/{instanceId}/Restart method is used to restart the workflow instance with the specified GUID. This can be used either after it has reached a failed state and the failure has been corrected (e.g. a CA was not responding when an enrollment was attempted or a PowerShell script failed to run to completion) or midstream while it's still active but in a suspended state waiting for signals to introduce a new version of the workflow definition. The workflow instance will restart from the beginning. This endpoint returns 204 with no content upon success.

Note: Only workflow instances with a Status of *Failed* or *Suspended* can be restarted.

Tip: The following permissions (see Security Roles and Claims on page 622) are required to use this feature:
/workflows/instances/manage/
/workflows/instances/read/

Table 825: POST Workflow Instances {instanceid} Restart Input Parameters

Name	In	Description
instanceId	Path	**Required**. A string indicating the Keyfactor Command reference GUID of the workflow instance to restart. Use the *GET /Workflow/Instances* method (see GET Workflow Instances on page 2655) to retrieve a list of all the workflow instances to determine the GUID. **Note:** When you restart an instance, it will be issued a new instance ID.
version	Body	An integer indicating the version number of the workflow definition. If no version is specified, the workflow will be restarted using the most recently published version.

Tip: See the *Keyfactor API Reference and Utility* which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation

for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

3.7 API Change Log

In this section you will find the change history for the Keyfactor API endpoints from version 9.0 onwards.

3.7.1 v9 API Change Logs

Find the change logs for Keyfactor API major release 9.0 and subsequent incremental releases below.

Release Type	Release Date	Link to Change Log
Major	August 2021	API Change Log v9.0 below
Incremental	September 2021	API Change Log v9.1 on page 2676
Incremental	October 2021	API Change Log v9.2 on page 2677
Incremental	November 2021	API Change Log v9.3 on page 2678
Incremental	December 2021	API Change Log v9.4 on page 2678
Incremental	January 2022	API Change Log v9.5 on page 2678
Incremental	February 2022	API Change Log v9.6 on page 2679
Incremental	March 2022	API Change Log v9.7 on page 2679
Incremental	April 2022	API Change Log v9.8 on page 2679
Incremental	May 2022	API Change Log v9.9 on page 2679

3.7.1.1 API Change Log v9.0

API changes for Keyfactor Command version 9.0 Major release

Table 826: API Change Log v9.0

Endpoint	Method	Action	Notes
/Agents/Approve	POST	Add	
/Agents/Disapprove	POST	Add	
/CertificateCollections	PUT	Add	
/CertificateCollections/Copy	POST	Add	
/Certificates/{id}/History	GET	Add	
/Certificates/{id}/Security	GET	Add	
/Certificates/{id}/Validate	GET	Add	
/Certificates/Locations/{id}	GET	Add	
/Certificates/Metadata/Compare	GET	Add	
/Certificates/Metadata/All	PUT	Add	
/Certificates/RevokeAll	POST	Add	
/CertificateStoreContainers	GET	Add	
/CertificateStoreContainers/{id}	GET	Add	
/CertificateStores/Certificates/Add	POST	Add	
/CertificateStores/Certificates/Remove	POST	Add	
/Enrollment/CSR/Context/My	GET	Add	
/Enrollment/PFX/Context/My	GET	Add	
/JobTypes/Custom	GET, POST, PUT	Add	
/JobTypes/Custom/{id}	GET, DELETE	Add	
/OrchestratorJobs/Custom	POST	Add	
/OrchestratorJobs/JobHistory	GET	Add	
/OrchestratorJobs/JobStatus/Data	GET	Add	
/Reports	GET, PUT	Add	

Endpoint	Method	Action	Notes
/Reports/{id}	GET	Add	
/Reports/{id}/Parameters	GET, PUT	Add	
/Reports/{id}/Schedules	GET, POST, PUT	Add	
/Reports/Custom	GET, POST, PUT	Add	
/Reports/Custom/{id}	GET, DELETE	Add	
/Reports/Schedules/{id}	GET, DELETE	Add	
/Security/Identities	GET, POST	Add	
/Security/Identities/{id}	DELETE	Add	
/Security/Identities/Lookup	GET	Add	
/Security/Roles	GET, POST, PUT	Add	
/Security/Roles/{id}	GET, DELETE	Add	
/SSH/Keys/Unmanaged	DELETE	Add	
/SSH/ServiceAccounts	DELETE	Add	
/SSH/Users/Access	POST	Add	
/SSL/Networks/{id}/Scan	POST	Add	

3.7.1.2 API Change Log v9.1

API changes for Keyfactor Command version 9.1 incremental release

Table 827: API Change Log v9.1

Endpoint	Methods	Action	Notes
/CertificateStores/{id}/Inventory	GET	Add	
/Enrollment/PFX/Replace	POST	Fix	SuccessfulStores collection now only includes Ids of stores that were successfully processed.
/Enrollment/PFX/Deploy	POST	Update	Now allows for multiple stores of the same type with different parameters.
/CertStoreTypes	POST/PUT	Update	EntryParameters can now be set via these methods.
/CertificateStores/Certificates/Add	POST	Update	Now allows for multiple stores of the same type with different parameters.
/CertificateStores/Certificates/Remove	POST	Update	Now allows for multiple stores of the same type with different parameters.
/CertificateCollections/{id}/Permissions	GET	Deprecate	

3.7.1.3 API Change Log v9.2

API changes for Keyfactor Command version 9.2 incremental release

Table 828: API Change Log v9.2

Endpoint	Methods	Action	Notes
/Certificates	GET	Fix	No longer fails if a collection id is not provided.
/OrchestratorJobs/JobHistory	GET	Fix	Request no longer fails for 'Dynamic' job types.
/Reports/Schedules/{id}	DELETE	Fix	Response code is now 200 when the user role does not have *Modify – Report* permission.

3.7.1.4 API Change Log v9.3

API changes for Keyfactor Command version 9.3 incremental release

Table 829: API Change Log v9.3

Endpoint	Methods	Action	Notes
/JobTypes/Custom	POST	Fix	No longer requires default field values.

3.7.1.5 API Change Log v9.4

API changes for Keyfactor Command version 9.4 incremental release

Table 830: API Change Log v9.4

Endpoint	Methods	Action	Notes
/Workflow/Certificates/Pending	GET	Update	Now returns the associated metadata.

3.7.1.6 API Change Log v9.5

API changes for Keyfactor Command version 9.5 incremental release

Table 831: API Change Log v9.5

Endpoint	Methods	Action	Notes
/Enrollment/PFX	POST	Update	No longer requires a certificate authority name to be provided.

3.7.1.7 API Change Log v9.6

API changes for Keyfactor Command version 9.6 incremental release.

No API endpoint changes were made in this release.

3.7.1.8 API Change Log v9.7

API changes for Keyfactor Command version 9.7 incremental release

Table 832: API Change Log v9.7

Endpoint	Methods	Action	Notes
/KeyfactorAPI/License	GET	Add	

3.7.1.9 API Change Log v9.8

API changes for Keyfactor Command version 9.8 incremental release.

No API endpoint changes were made in this release.

3.7.1.10 API Change Log v9.9

API changes for Keyfactor Command version 9.9 incremental release

Table 833: API Change Log v9.9

Endpoint	Methods	Action	Notes
/Reports/<any>	GET	Fix	Spaces within the sortField no longer results in an exception.
/Reports/{id}/Schedules	GET	Fix	An invalid sortField no longer results in an exception.
/Agents	GET	Update	New query parser to support the AgentId GUID.

3.7.2 v10 API Change Logs

Find the change logs for Keyfactor API major release 10.0 and subsequent incremental and hot fix releases below.

Release Type	Release Date	Link to Change Log
Major	September 2022	API Change Log v10.0 below
Incremental	November 2022	API Change Log v10.1 on page 2686
Incremental	January 2023	API Change Log v10.2 on page 2686
Hot Fix	April 2023	API Change Log v10.3.1 on page 2686
Incremental	May 2023	API Change Log v10.4 on page 2687
Hot Fix	July 2023	API Change Log v10.4.3 on page 2687
Hot Fix	September 2023	API Change Log v10.4.5 on page 2688
Hot Fix	September 2023	API Change Log v10.4.6 on page 2688

3.7.2.1 API Change Log v10.0

API changes for Keyfactor Command version 10.0 Major release

Table 834: API Change Log v10.0

Endpoint	Methods	Action	Notes
/Agents/{id}	GET	Add	
/Agents/Reset	POST	Add	
/AgentBlueprint	GET	Add	
/AgentBlueprint/{id}	GET, DELETE	Add	
/AgentBlueprint/{id}/Jobs	GET	Add	
/AgentBlueprint/{id}/Stores	GET	Add	
/AgentBluePrint/ApplyBlueprint	POST	Add	
/AgentBluePrint/GenerateBluePrint	POST	Add	
/Alerts/Denied	GET, PUT, POST	Add	
/Alerts/Denied/{id}	GET, DELETE	Add	
/Alerts/Expiration	GET, PUT, POST	Add	
/Alerts/Expiration/{id}	GET, DELETE	Add	
/Alerts/Expiration/Schedule	GET, PUT	Add	
/Alerts/Expiration/Test	POST	Add	
/Alerts/Expiration/TestAll	POST	Add	
/Alerts/IssuedAlerts	GET, PUT, POST	Add	
/Alerts/IssuedAlerts/{id}	GET, DELETE	Add	
/Alerts/Issued/Schedule	GET, PUT	Add	
/Alerts/KeyRotation	GET, PUT, POST	Add	

Endpoint	Methods	Action	Notes
/Alerts/KeyRotation/{id}	GET, DELETE	Add	
/Alerts/KeyRotation/Schedule	GET, PUT	Add	
/Alerts/KeyRotation/Test	POST	Add	
/Alerts/KeyRotation/TestAll	POST	Add	
/Alerts/Pending	GET, PUT, POST	Add	
/Alerts/Pending/{id}	GET, DELETE	Add	
/Alerts/Pending/Schedule	GET, PUT	Add	
/Alerts/Pending/Test	POST	Add	
/Alerts/Pending/Test/{id}	POST	Add	
/CertificateAuthorities	GET	Update	Schedules are now included in the results.
/CertificateAuthorities	POST	Update	Ability to turn off schedules, sessions are abandoned properly, and threshold monitoring schedule is included.
/CertificateAuthorities/{id}	PUT	Update	Ability to turn off schedules, sessions are abandoned properly, and threshold monitoring schedule is included.
/CertificateAuthorities/{id}	DELETE	Update	Deletion is now prevented if schedules are associated.
/CertificateCollections	POST	Update	Query parameter no longer needed when a valid CopyFromId is provided.
/CertificateCollections/{id}/Permissions	POST	Deprecated	Replaced by /Security/Roles/{id}/Permissions/Collection.
/Certificates/Analyze	POST	Add	
/Certificates/IdentityAudit/{id}	GET	Add	

Endpoint	Methods	Action	Notes
/CertificateStoreContainers	POST	Add	
/CertificateStoreContainers/{id}	PUT, DELETE	Add	
/CertificateStores/Server	GET, POST, PUT	To Be Deprecated	Server usernames, server passwords, and the UseSSL flag are managed by the /CertificateStores API endpoints directly as JobProperties using the Properties parameter, replacing the deprecated /CertificateStores/Server API endpoints.
/CertificateStores	GET, POST, PUT	Updated	Server usernames, server passwords, and the UseSSL flag are managed by the /CertificateStores API endpoints directly as JobProperties using the Properties parameter, replacing the deprecated /CertificateStores/Server API endpoints.
/Enrollment/PFX (v2)	POST	Add	
/Enrollment/Settings/{id}	GET	Add	
/JobTypes/Custom	POST	Update	DefaultValue property is no longer required, validation is now performed on the JobTypeFields/DefaultValue property, validation prevents names containing spaces.
/JobTypes/Custom/{id}	DELETE	Update	Includes validation so that deletion is prevented if at least one associated approved orchestrator implements the capability.
/MacEnrollment	GET, PUT	Add	
/Monitoring/Revocation	GET, POST	Update	Renamed from /Workflow/RevocationMonitoring

Endpoint	Methods	Action	Notes
/Monitoring/Revocation/{id}	GET, PUT, DELETE	Update	Renamed from /Workflow/RevocationMonitoring/{id}
/Monitoring/Revocation/Test	POST	Add	
/Monitoring/Revocation/TestAll	POST	Add	
/Orchestrators/JobHistory	GET	Update	Added JobId field.
/Orchestrators/ScheduledJobs	GET	Add	
/OrchestratorJobs/Reschedule	POST	Add	
/OrchestratorJobs/Unschedule	POST	Add	
/OrchestratorJobs/Acknowledge	POST	Add	
/Security/Identities/{id}	GET	Add	
/Security/Roles/{id}/Identities	GET, POST	Add	
/Security/Roles/{id}/Containers	GET, POST	Add	
/Security/Roles/{id}/Copy	POST	Add	
/Security/Roles/{id}/Permissions	GET	Add	
/Security/Roles/{id}/Permissions/Global	GET, POST, PUT	Add	
/Security/Roles/{id}/Permissions/Collections	GET, POST, PUT	Add	Replaced the /CertificateCollections/{id}/Permissions endpoint functionality.
/Security/Roles/{id}/Permissions/Containers	GET, POST, PUT	Add	Returns only containers that have a permission set for the selected security role.
/SMTP	GET, PUT	Add	
/SMTP/Test	POST	Add	
/Templates	GET, PUT	Update	Includes template-specific policy information.
/Templates/{id}	GET	Update	Includes template defaults.

Endpoint	Methods	Action	Notes
/Templates/Settings	GET, PUT	Update	Includes global template policies.
/Template/SubjectParts	GET	Add	
/Templates/Global/Settings	GET, PUT	Add	
/Templates/Import	POST	Add	
/Workflow/Certificates/Pending	GET	Update	Now supports query fields of Requester and RequestType.
/Workflow/Definitions/Steps/{extensionName}	GET	Add	
/Workflow/Definitions/{definitionId}	GET, PUT, DELETE	Add	
/Workflow/Definitions	GET, POST	Add	
/Workflow/Definitions/Steps	GET	Add	
/Workflow/Definitions/Types	GET	Add	
/Workflow/Definitions/{definitionId}/Steps	PUT	Add	
/Workflow/Definitions/{definitionId}/Publish	POST	Add	
/Workflow/Instances/{instanceId}	GET, DELETE	Add	
/Workflow/Instances	GET	Add	
/Workflow/Instances/My	GET	Add	
/Workflow/Instances/AssignedToMe	GET	Add	
/Workflow/Instances/{instanceId}/Stop	POST	Add	
/Workflow/Instances/{instanceId}/Signals	POST	Add	
/Workflow/Instances/{instanceId}/Restart	POST	Add	

3.7.2.2 API Change Log v10.1

API changes for Keyfactor Command version 10.1 incremental release

Table 835: API Change Log v10.1

Endpoint	Methods	Action	Notes
/Templates	PUT, GET	Update	Ed448 and Ed25519 keys are now supported for certificate enrollment, policy, import and search.
/Templates/{id}	GET	Update	Ed448 and Ed25519 keys are now supported for certificate enrollment, policy, import and search.
/Templates/Settings	PUT, GET	Update	Ed448 and Ed25519 keys are now supported for certificate enrollment, policy, import and search.

3.7.2.3 API Change Log v10.2

API changes for Keyfactor Command version 10.2 incremental release

Table 836: API Change Log v10.2

Endpoint	Methods	Action	Notes
/Security/My	GET	Add	Returns all the security roles and global permissions for the requesting user.
/Enrollment/CSR	POST	Update	The workflow instance ID has been added to the response.
/Enrollment/CSR	POST	Update	A new PrivateKey input field has been added to support private key retention on CSR enrollment.
/Enrollment/PFX	POST	Update	The workflow instance ID has been added to the response.
/Certificates/Analyze	POST	Update	The endpoint requires Global Certificates-Read or Certificates-Import permissions.

3.7.2.4 API Change Log v10.3.1

API changes for Keyfactor Command version 10.3.1 hot fix release

Table 837: API Change Log v10.3.1

Endpoint	Methods	Action	Notes
/Reports/{id}/Schedules	POST	Fixed	Reports can be scheduled when the user scheduling the report only has permission to view one certificate collection.

3.7.2.5 API Change Log v10.4

API changes for Keyfactor Command version 10.4 incremental release

Table 838: API Change Log v10.4

Endpoint	Methods	Action	Notes
/Enrollment/CSR	POST	Fixed	Includes SANs entered outside the CSR only when the *Allow CSR SAN Entry* application setting is set to true. SANs entered outside the CSR replace SANs in the CSR rather than appending to SANs from the CSR.
/Workflow/Instances	GET	Fixed	Inclused SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/AssignedToMe	GET	Fixed	Inclused SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/My	GET	Fixed	Inclused SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/{instanceId}	GET	Fixed	Inclused SANs entered outside the CSR in workflow instance details.

3.7.2.6 API Change Log v10.4.3

API changes for Keyfactor Command version 10.4.3 hot fix release

Table 839: API Change Log v10.4.3

Endpoint	Methods	Action	Notes
/CertificateAuthority/Test	POST	Fixed	EJBCA version 8 is supported.
/Enrollment/Renew	POST	Fixed	EJBCA version 8 is supported.
/Templates/Import	POST	Fixed	EJBCA version 8 is supported.

3.7.2.7 API Change Log v10.4.5

API changes for Keyfactor Command version 10.4.5 hot fix release

Table 840: API Change Log v10.4.5

Endpoint	Methods	Action	Notes
/CSRGerneration/Generate	POST	Update	3072-bit RSA keys are supported.
/Enrollment/CSR	POST	Update	3072-bit RSA keys are supported.
/Enrollment/PFX	POST	Update	3072-bit RSA keys are supported.
/Enrollment/Renew	POST	Update	3072-bit RSA keys are supported.

3.7.2.8 API Change Log v10.4.6

API changes for Keyfactor Command version 10.4.6 hot fix release

Table 841: API Change Log v10.4.6

Endpoint	Methods	Action	Notes
/Enrollment/CSR	POST	Fixed	Includes SANs entered outside the CSR only when the *Allow CSR SAN Entry* application setting is set to true. SANs entered outside the CSR replace SANs in the CSR rather than appending to SANs from the CSR.
/Workflow/Instances	GET	Fixed	Includes SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/AssignedToMe	GET	Fixed	Includes SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/My	GET	Fixed	Includes SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/{instanceId}	GET	Fixed	Includes SANs entered outside the CSR in workflow instance details.

3.7.3 v11 API Change Logs

Find the change logs for Keyfactor API major release 11.0 below.

Release Type	Release Date	Link to Change Log
Major	October 2023	API Change Log v11.0 below

3.7.3.1 API Change Log v11.0

API changes for Keyfactor Command version 11.0 major release

Table 842: API Change Log v11.0

Endpoint	Methods	Action	Notes
AppSetting	GET, PUT	Added	
AppSetting/{id}	GET	Added	
AppSetting/{id}/Set	PUT	Added	
AppSetting/{name}/Set	PUT	Added	
CertificateAuthority/SourceCount	GET	Added	
CertificateAuthority/ConfigurationTenants	GET	Added	
CertificateAuthority/HealthMonitoring/Schedule	GET	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients	GET	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients	POST	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients	GET	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients	POST	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients/{id}	DELETE	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients/{id}	GET	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients/{id}	PUT	Added	

Endpoint	Methods	Action	Notes
CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id}	DELETE	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id}	GET	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id}	PUT	Added	
CertificateAuthority/Import	POST	Added	
CertificateAuthority/ConfigurationTenants	GET	Changed	The endpoint is now renamed to GET /CertificateAuthority/AvailableForests and the definition is changed to: Returns a list of available forests that are in Active Directory.
Certificates/CSV	GET	Added	
Certificates/IdentityAudit/{id}	GET	Added to V2 definitions	This API endpoint is available in both the V1 and V2 definitions in the Keyfactor API Reference and Utility and acts exactly the same in both.
CertificateCollections/{id}/Permissions	POST	Removed	Instead use POST Security/Roles/{id}/Permissions/Collection.
CertificateCollections/{id}	DELETE	Added	
CertificateCollections/NavItems	GET	Added	

Endpoint	Methods	Action	Notes
CertificateCollections/CollectionList	GET	Added	
CertificateCollections/{id}/Favorite	PUT	Added	
CertificateStores/Server	GET, POST, PUT	Deprecated	
CertificateStoreTypes	GET	Changed	The API will return ALL certificate store types if at least one of these conditions are met: • The end-user has one of the /certificate_stores/read/ global permissions. • The end-user has permission to at least one certificate store container.
ComponentInstallation/{id}	DELETE	Added	
ComponentInstallation/	GET	Added	
EventHandlerRegistration/{id}	GET, DELETE, PUT	Added	
EventHandlerRegistration/	GET, POST	Added	
Extensions/Scripts/{id}	DELETE, GET	Added	
Extensions/Scripts	GET, POST, PUT	Added	
IdentityProviders/{id}	GET, PUT	Added	
IdentityProviders	GET	Added	
IdentityProviders/Types	GET	Added	

Endpoint	Methods	Action	Notes
Permissions	GET	Added	
PermissionSets/{id}	GET, DELETE	Added	
PermissionSets	GET, POST, PUT	Added	
Scheduling	POST	Added	
Security/Containers/{id}/Roles	GET, POST	Added	
Security/Audit/Collections/{id}	GET	Added	
Security/Claims/{id}	GET, DELETE	Added	
Security/Claims	GET, POST, PUT	Added	
Security/Claims/Roles	GET	Added	
Security/Identities	GET	Changed	The non-working query string field has been removed.
Security/Roles/{id}/Permissions/PamProviders	GET, PUT	Added	
Security/Roles (V1) Security/Roles/{id} (V1) Security/Roles/{id}/Identities(V1) Security/Roles/{id}/copy(V1)	GET, POST, PUT	Deprecated in V1	All SecurityRoles API endpoints (except DELETE /{id}) have been deprecated from the V1 API, as they only work against Active Directory users. There are new Security/Roles endpoints in the V2 API
Security/Roles(V2) Security/Roles/{id}(V2)	GET, POST, PUT	Added in V2	Security/ Roles API endpoints have been recreated in V2 API to work with both OAUTH and AD users.

Endpoint	Methods	Action	Notes
Templates/{id}	GET	Changed	Now returns an object with a TemplatePolicy property and a KeyAlgorithms property that show the policies and algorithms the template supports.
Templates/Import	GET, POST	Changed	Now supports multiple algorithms.
Templates/Settings	GET, PUT	Changed	The Template Policy property used to update global application settings now contains four properties: ECDSA, RSA, Ed448, and Ed25519. These replace the AllowEd448, AllowEd25519, RSAValidCurves, and ECCValidCurves.

4.0 Installing Servers

The Keyfactor Command solution by Keyfactor allows you to issue and manage certificates across enterprise infrastructures to allow you to achieve end-to-end visibility, control, and automation across all your machine identities so you can turn the impossible into the possible. It includes a web-based Management Portal running on a SQL backend providing the command and control center for managing certificates in the enterprise.

Keyfactor Command provides:

- Visibility

  Identify risks and prevent outages more effectively with a complete and continuous inventory of all your cryptographic assets.

- Control

  Have ultimate flexibility to make all certificates trusted, compliant, and up-to-date—and keep them that way.

- Automation

  Replace manual, error-prone tasks with automated key and certificate discovery, management, and renewal.

- Orchestration

  Move from DevOps to DevSecOps by orchestrating and expanding cryptography to secure software delivery pipelines.

In addition to the Management Portal, Keyfactor also offers:

- Several agents and orchestrators for managing certificates in certificates stores via the Management Portal (see Installing Orchestrators on page 2875).
- Several certificate authority gateways to support management of and enrollment for certificates from remote and cloud-based certificate providers via the Management Portal.
- The Keyfactor API that integrates with the product to provide for customization (see Keyfactor API Reference on page 843).
- A certificate authority policy module with several policy handlers to provide policy control at the Microsoft CA level (see Keyfactor CA Policy Module on page 2844).
- An SSH Key Manager that extends beyond certificate management and traditional PKI to give security and network teams a simple, centralized solution to discover and manage SSH keys across their server and cloud infrastructure (see SSH on page 525).

Uniquely designed for PKI administrators to operate an enterprise PKI, it's never been easier to issue, revoke, renew, or replace a digital certificate. With exceptionally robust reporting and management capabilities for all the certificates in an IT environment, the PKI administrator has a truly scalable and entirely secure system for operating an enterprise PKI.

4.1 Logical Architecture

Keyfactor Command is an n-tier application, consisting of a web/presentation layer, application tier, and database tier. In addition, Keyfactor Command optionally includes a number of enrollment and management components to help facilitate secure and/or automated certificate issuance and delivery to various server and client platforms. The following sections provide views of the Keyfactor Command architecture from a logical and physical standpoint.

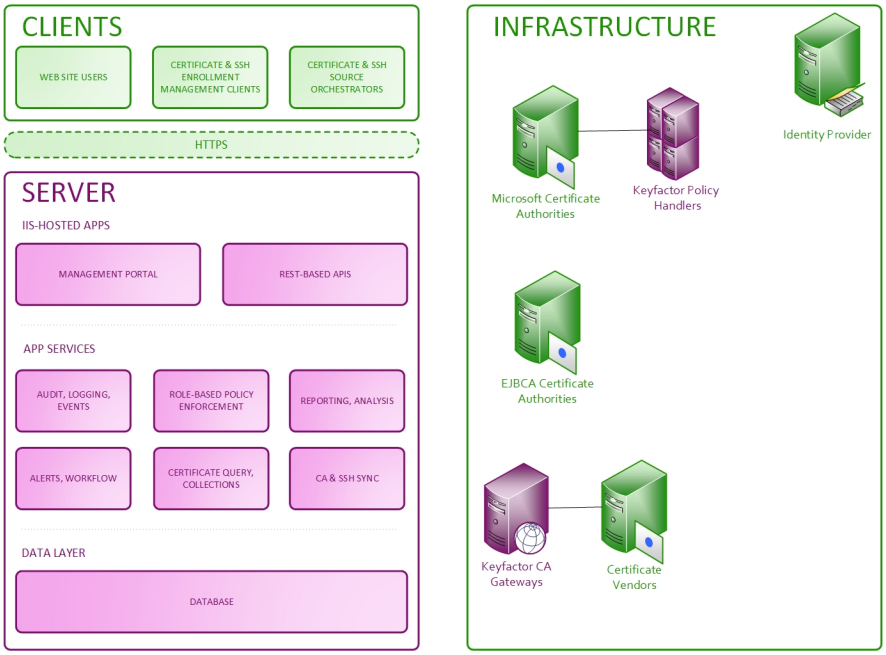


Figure 441: Keyfactor Command Logical Architecture Diagram

The Keyfactor Command solution includes the following logical components:

- Client / Orchestrator Tier:
  - Certificate Enrollment and Management Tools—While many certificate management functions can be performed in a completely agentless fashion, Keyfactor Command provides a number of enrollment and management tools to enable enhanced functionality where needed.
  - Certificate Source Orchestrators (aka Agents) and Gateways—Keyfactor Command gathers information about an enterprise's certificates and SSH keys from a number of different sources, including Microsoft and EJBCA CA databases, SSL scans, SSH key scans, API-based import, Java keystores, PEM certificate stores, F5 devices, NetScaler devices, Amazon Web Services (AWS) locations, select certificate vendor certificates via gateways, and manual import through the Keyfactor Command Management Portal.
- Web Tier:
  - Management Portal—Keyfactor Command includes a web-based Management Portal that provides a PKI operations dashboard for administrators. It also enables certificate officers to

easily search for and locate certificates and then perform management functions on them such as revocation or recovery. In addition, Keyfactor Command allows every certificate to be tagged with additional customer-defined metadata about the certificate, such as points of contact, certificate/app owners, etc. From within the Management Portal, administrators can inventory and manage secure shell (SSH) keys across the enterprise, while users can issue new SSH keys.

- Enrollment Web Pages—Keyfactor Command includes issuance capabilities to a wide array of platforms, including Mac auto-enrollment, PKCS#12-based certificate issuance, and web-based CSR submission for administrator enrollment. PKCS#12 (PFX) and CSR enrollment are supported against Microsoft CAs in the local forest, remote CAs—with or without trust relationships—and non-domain-joined Microsoft and EJBCA CAs.
- Web APIs—Keyfactor Command is implemented with a robust and continually growing set of APIs that allow integration of Keyfactor Command functionality with the set of Keyfactor Command clients and orchestrators, as well as third-party or customer-created software or scripts.

- App Tier:
  - Event History and Audit Logging—Keyfactor Command maintains a record of operations that are performed on a certificate and the individual who performed the operation. This includes information such as initial synchronization date, additions to and removals from certificate stores, certificate recovery, and certificate revocation.
  - Role-Based Policy Enforcement—Keyfactor Command offers a rich, role-based permissions model that allows you to create your own roles as needed within the Keyfactor Command Management Portal. Users can be assigned to roles based upon Active Directory group memberships or individually, and then each role can be assigned granular Keyfactor Command permissions such as report creation, certificate revocation or renewal, or metadata update.
  - Dashboard and Report Engine—Keyfactor Command contains a dynamic dashboard along with several built-in reports generated using the Logi Analytics Platform.
  - Certificate Query & Collections—Keyfactor Command allows certificate administrators to query the certificate database using various search criteria. In addition, the bulk of Keyfactor Command's reporting and automated notification functionality can be driven through certificate collections, which are a user-definable mechanism that allows organizations to report on groups of certificates based on selection criteria.
  - Workflow Builder—The workflow builder in Keyfactor Command allows you to easily automate event-driven tasks when a certificate is requested or revoked. The workflows can be configured with multiple steps between the start and end of the operation that offer a simple way to configure notifications, approvals, and end-to-end automation. This provides for operational agility in an intuitive and easy-to-configure manner. The workflow builder is highly customizable with options to execute PowerShell scripts, invoke REST requests, send email messages, and require one or more approvals built in, and facilities to build custom steps to allow many more functions to be built as needed.
  - Alert Notice Generator—Keyfactor Command allows you to configure customized email notifications for impending certificate expiration, revocation expiration, pending certificate requests, issued certificate requests and denied certificate requests. These notifications

can be sent at configurable intervals, and may contain ASCII or HTML content, along with relevant information about the certificate or request in question (e.g. subject DN, issuer, thumbprint, template, custom metadata, etc.)

    - Certificate Request Alerting—Keyfactor Command provides interfaces through which administrators can request certificates that require CA-level manager approval, interfaces where the approvers can either issue or deny the certificate request, and interfaces where the requesters can then download the certificates. This, along with the notice generator, provides an end-to-end flow for certificate requests that require CA-level manager approval.
    - Alert Handlers—In addition to the notice generator that provides email alerts for SSH key and certificate expiration and enrollment workflow, Keyfactor Command also provides optional handlers that can be used in the certificate request and expiration alerts to output the information to the event log rather than sending it via email, run a PowerShell script, or automatically renew expiring certificates that are found in certificate store locations.
    - Keyfactor Command Service—The Keyfactor Command Service (a.k.a. the timer service) is designed to continually keep the Keyfactor Command SQL database synchronized with the contents of every configured Microsoft and EJBCA CA database in the organization as well as external certificates located on servers it can scan. The service can perform full or partial scans of different CAs at user-defined intervals. This enables a rapidly-accessible, easily queried mirror of CA database information that can then be put to use via Keyfactor Command. Synchronization of CA information is supported for Microsoft CAs joined to the local forest, remote domain-joined Microsoft CAs—with or without trust relationships—and non-domain-joined Microsoft and EJBCA CAs. The Keyfactor Command Service is also responsible for executing a variety of periodic tasks, including scheduled reports, alerts and cleanup jobs.

- Data Tier:
    - SQL Database—Keyfactor leverages a Microsoft SQL Server database to store the information that Keyfactor Command uses.
- Microsoft Certification Authority Components:
    - RFC 2818 Policy Handler—The RFC 2818 Policy Handler integrates with the Microsoft CA to allow you to automate the addition of a DNS SAN matching the CN of the requested certificate for selected templates.
    - SAN Attribute Policy Handler—The SAN Attribute Policy Handler allows the addition of SANs not included in the CSR when making a CSR enrollment request. The added SANs will overwrite any existing SANs in the CSR. This functionality is the same as that seen with the Microsoft default policy module for the CA as a whole when the CA EDITF_ATTRIBUTESUBJECTALTNAME2 flag is set except the SAN Attribute Policy Handler provides the ability to control SAN addition on a template-by-template basis without the need to enable the Microsoft CA EDITF_ATTRIBUTESUBJECTALTNAME2 flag.
    - Whitelist Policy Handler—The Whitelist Policy Handler integrates with the Microsoft CA to allow you to restrict certificate enrollment on that CA for a configured certificate template or templates to only designated client machines. This allows you, for example, to force certificate enrollment for web server certificates to be accepted only via the Keyfactor Command

Management Portal and denied when coming from the Microsoft certificates MMC or IIS on the target servers for web server certificates.

- Enterprise Infrastructure:
  - Certification Authorities—Keyfactor Command has been built from the ground up to make it easier to operate organizational PKIs. This allows you to benefit from Keyfactor Command's extended features around Microsoft CA capabilities such as certificate templates, enrollment and recovery agents, and private key recovery. Keyfactor Command's integration with EJBCA provides support for capabilities such as certificate profiles, end entity profiles, enrollment, and revocation.
  - Identity Providers—Keyfactor Command relies on an identity provider to support authentication to the Management Portal and the Keyfactor API and supporting group memberships for Keyfactor Command role assignments. Historically, the product has been integrated with Microsoft Active Directory, using AD for Microsoft CA and certificate template enumeration and for the inclusion of AD account attributes in the content of issued certificates. With the release of Keyfactor Command version 11.0, support for identity providers expands beyond Active Directory. Users may choose between Active Directory and an open authorization (OAuth) 2.0 compliant identity provider with a complete implementation of the OpenID Connect (OIDC) protocol including the Keyfactor-provided Keyfactor Identity Provider.

4.2 Physical Architecture

Figure 442: Keyfactor Command Physical Architecture Diagram shows the physical architecture of the Keyfactor Command solution.

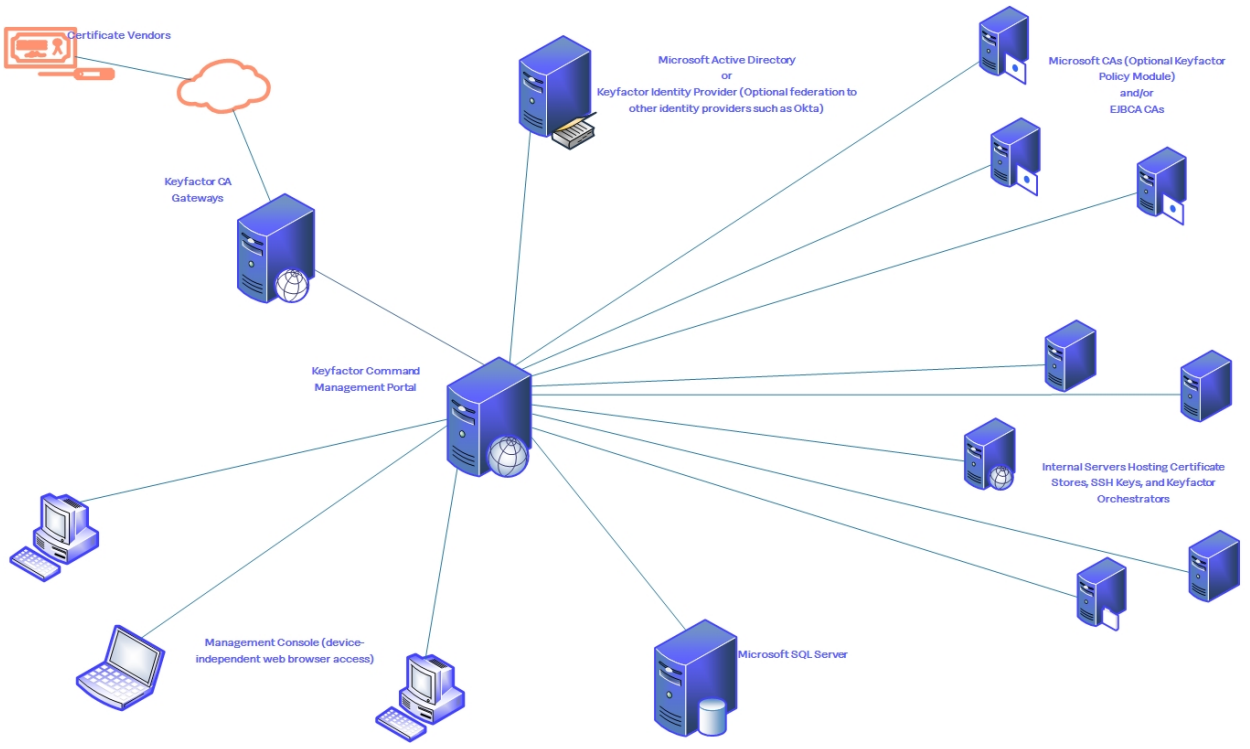


Figure 442: Keyfactor Command Physical Architecture Diagram

For simplicity, the servers in Figure 442: Keyfactor Command Physical Architecture Diagram are shown as single physical instances. In practice, these servers may be virtual machines and may be load balanced or clustered to meet availability or performance requirements. The diagram includes some optional components—including the Keyfactor vendor gateways and Keyfactor orchestrators—which are not covered in this guide. For more information about these components, see the Installing Orchestrators on page 2875 guide and the documentation for each of the gateways.

- Keyfactor Command-Dedicated Servers[1]:
  - Keyfactor Command Server—This server hosts the Keyfactor Command Management Portal, the Keyfactor Command Services roles, and the Logi Analytics Platform for report generation. These roles run as ASP.NET (4.5 or higher) applications on IIS. Both Windows Server 2019 and 2022 are supported.
- Enterprise-Shared Servers:
  - Microsoft SQL Server—Keyfactor Command supports Microsoft SQL Server 2017, 2019 and 2022 all with TLS encryption enabled for its primary database. While a dedicated SQL deployment is certainly an option, many organizations maintain a well-established SQL server farm to support multiple applications within the organization; if preferred, Keyfactor Command can easily make use of such a service. Keyfactor does not recommend locating the Keyfactor Command roles on the SQL server in a production deployment.

[1]The roles described in this section may be co-located on a single physical or virtual server or may be further separated to multiple machines.

- Web Reverse Proxy—If Internet-based access is required, the Keyfactor Command services can be published through a variety of reverse proxy products such as Microsoft UAG/TMG, F5, SiteMinder, or Citrix NetScaler.
- Network-based Hardware Security Module (HSM not pictured)—In certain configurations, Keyfactor Command requires the use of Enrollment Agent (EA) and/or Key Recovery Agent (KRA) certificates. To provide additional security over these certificates' private keys, Keyfactor strongly recommends the use of a Hardware Security Module (HSM) such as the Thales NetHSM if these features will be used.

4.3 Solution Design

Keyfactor Command supports a number of different deployment architectures to help provide for different needs from small and simple to highly available. The solution can be as simple as one Keyfactor Command server hosting all the Keyfactor Command roles (other than the policy handlers, which are installed on a Microsoft CA) or the roles can be separated onto different machines to provide increased security or distribute the load. Redundant servers can be added to provide for high availability—either within the same data center or across data centers. Keyfactor expects that the specifics of a high availability deployment plan would be finalized as part of the project rollout.

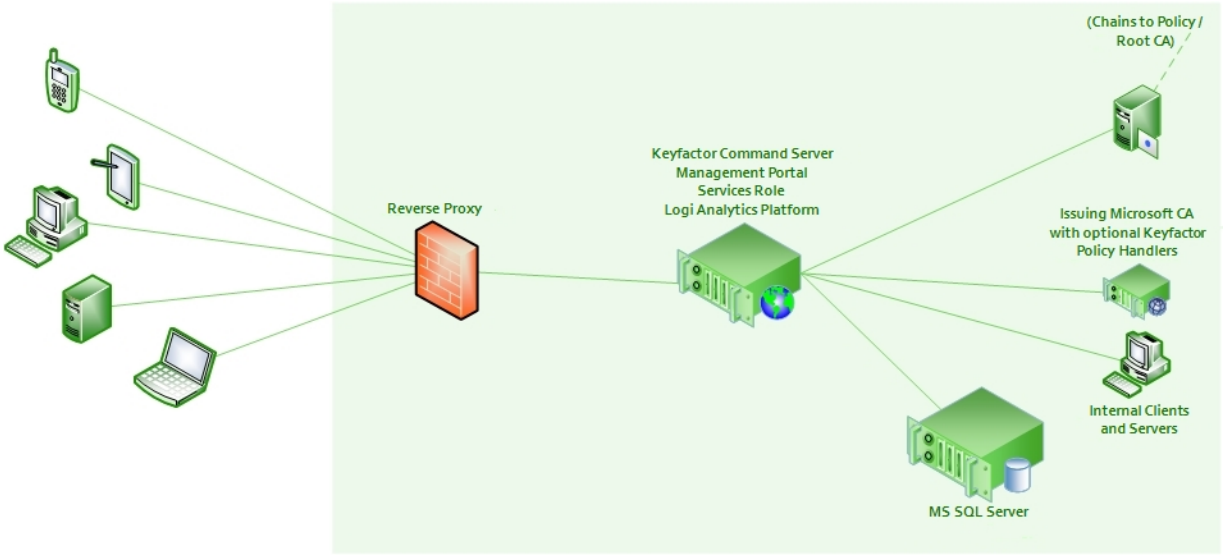


Figure 443: Simple Keyfactor Command Solution Design

4.4 Keyfactor Command Server

The Keyfactor Command solution by Keyfactor allows you to issue and manage certificates across enterprise infrastructures to allow you to achieve end-to-end visibility, control, and automation across all your machine identities so you can turn the impossible into the possible.

4.4.1 System Requirements

Table 843: System Requirements provides the recommendations for minimum system specifications used by Keyfactor Command components. All servers may be deployed as virtual machines and may be part of a clustering or load-balanced architecture, if desired. If the Keyfactor Command roles are co-located, the specifications may need to be scaled accordingly. All Microsoft-supported methods for making SQL Server highly available are supported. For most high availability requirements, Keyfactor recommends using always on availability groups (see SQL Server on page 2742).

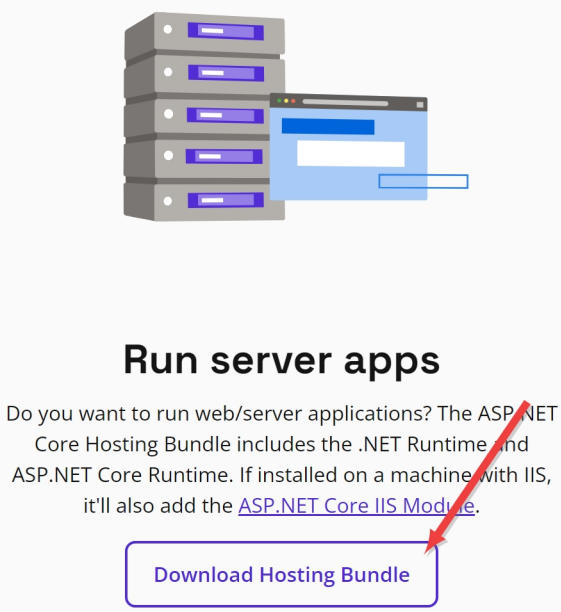
Important: SSH management in Keyfactor Command with the Keyfactor Bash Orchestrator (see SSH on page 525) is only supported when using Active Directory as an identity provider (see Selecting an Identity Provider for Keyfactor Command on page 2704). The SSH option in the Management Portal will only appear when Keyfactor Command is installed using Active Directory as an identity provider (and with a license that supports SSH).

Table 843: System Requirements

Component	Minimum Requirements

Keyfactor Command Server (Management Portal, Keyfactor API, and Services roles)	• Windows Server 2019 or 2022
	• Internet Information Services (IIS) with:
	  • Basic Authentication—if you plan to use Active Directory as an identity provider (see Selecting an Identity Provider for Keyfactor Command on page 2704)
	  • Windows Authentication—if you plan to use Active Directory as an identity provider with Windows authentication (see Selecting an Identity Provider for Keyfactor Command on page 2704)
	  • ASP.NET 4.7 or greater
	  • The Active Directory Module for Windows PowerShell
	See Install IIS and .NET on the Keyfactor Command Server on page 2768.
	• ASP.NET Core Hosting Bundle version 6.0 (x64). Version 6.0 is available for download from Microsoft:
	https://dotnet.microsoft.com/download/dotnet/6.0/runtime
	You need the ASP.NET Core Hosting Bundle, not the .NET Runtime (x64) or the ASP.NET Core Runtime. At the above link, this would be the **Download Hosting Bundle** option under the *Run server apps* heading.

Component	Minimum Requirements

	
	Figure 444: Select the Download Hosting Bundle Option Under Run Server Apps
	You can use the following PowerShell command to check the .NET core version (s) installed on a server (if any):
	`dotnet --list-runtimes`
	Output from this command will look something like this if you have the correct 6.0 x64 version of the .NET Hosting Bundle installed (notice the path is in C:\Program Files, not C:\Program Files (x86), indicating this is the x64 version):
	`Microsoft.AspNetCore.App 6.0.21 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]`
	⚠ **Important:** The ASP.NET Core Hosting Bundle should not be installed before installing IIS. If the hosting bundle is installed before IIS is installed, the bundle will not function correctly after the IIS install and will require repair.
	• .NET Framework 4.7.2 or greater
	• 4 GB RAM, 2 GHz CPU, 40 GB disk
	• Keyfactor Command license key for the current release

Microsoft SQL Database	Ability to connect to a Microsoft SQL Server 2017, 2019, or 2022 all with TLS encryption enabled and compatibility level 130 or higher. 8 GB RAM, 2+ GHz CPU (>= 2 cores), 500 GB disk

Component	Minimum Requirements
Browser to Access the Management Portal	• Chrome: 99.0.4844.74+ • Firefox: 98.0+ • Microsoft Edge: 99.0.1150.30+
EJBCA CA (Optional)	• EJBCA Enterprise version 7.8.1 or later is supported. • The EJBCA REST API must be enabled to interoperate with Keyfactor Command (see System Configuration -> Protocol Configuration in the EJBCA administration portal).

4.4.2 Planning & Preparing

Before you install Keyfactor Command, you need to consider the components that make it up and its dependencies and decide where you want each role to reside, which roles—if any—you want to be highly available, and which features you're going to enable. It's possible to start with a non-redundant implementation and then add redundancy at a later time, but it's best to plan ahead for this if it's the desired goal.

Your license for Keyfactor Command may not include all the roles described in this document, so some sections of this guide may not apply to your implementation.

Once you've made these planning decisions, you then need to follow the steps outlined in this section that need to be taken prior to a Keyfactor Command implementation to install the prerequisites, create the required supporting components, and gather the necessary information to complete the Keyfactor Command installation and configuration process.

4.4.2.1 Selecting an Identity Provider for Keyfactor Command

Identity providers are used to provide a method for authenticating access to Keyfactor Command. Keyfactor Command supports Microsoft Active Directory and open authorization (OAuth) 2.0 compliant identity providers with a complete implementation of the OpenID Connect (OIDC) protocol. Keyfactor Command has been tested with the following identity providers:

- Active Directory

  Microsoft's Active Directory has historically been the only identity provider supported by Keyfactor Command. With Active Directory, you can authenticate users defined in the Active Directory forest to which the Keyfactor Command server is joined and users from forests in a trust with this forest using integrated Windows authentication. Users may alternatively be authenticated to Keyfactor Command using Basic authentication when you opt for Active Directory as your identity provider. Active Directory supports user, group and computer accounts.
- Keyfactor Identity Provider

Keyfactor Identity Provider is a lightweight application that is easily installed in the same environment as Keyfactor Command to provide standalone authentication separate from Active Directory. It may be used directly to supply authentication or it may be used to federate authentication to another OAuth 2.0 compliant identity provider (e.g. Okta, Ping Identity). Keyfactor Identity Provider runs in a Linux-based Docker container. Keyfactor Identity Provider supports users and groups.

- Auth0

  Auth0 is a cloud-based OAuth 2.0 compliant identity and access management (IAM) solution owned by Okta.

A given Keyfactor Command server may be configured with only one identity provider. If desired, you may configure an environment with multiple Keyfactor Command servers and configure a different identity provider for each Keyfactor Command server.

Important: SSH management in Keyfactor Command with the Keyfactor Bash Orchestrator (see SSH on page 525) is only supported when using Active Directory as an identity provider. The SSH option in the Management Portal will only appear when Keyfactor Command is installed using Active Directory as an identity provider (and with a license that supports SSH).

Installing Keyfactor Identity Provider

If you've opted to use Keyfactor Identity Provider as your identity provider for Keyfactor Command, you'll need to install and configure it before installing Keyfactor Command. Keyfactor Identity Provider runs in a Docker container on a Linux machine and is configured to use a Microsoft SQL database to store its data (configuration, user and group accounts, etc).

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

System Requirements

Keyfactor Identity Provider has the following requirements:

- Linux
- Docker
- Java version 12 or later
- Microsoft SQL Server

This guide assumes you are starting from a base of already having a Linux server with Docker installed. Instructions for installing and configuring Linux and Docker are beyond the scope of this guide. Some helpful web pages include:

- https://ubuntu.com/server/docs/installation
- https://docs.docker.com/engine/install/ubuntu/

Preparing

To get ready to install Keyfactor Identity Provider you will need to gather a few pieces of information, copy some certificate files to your Linux machine, and set up a database and user in SQL.

Prepare Certificates

Keyfactor Identity Provider uses two certificates:

- The public key certificate of the Microsoft SQL server that will host the database for Keyfactor Identity Provider to allow it to connect to the SQL server using an encrypted connection (see Using SSL to Connect to SQL Server on page 2746).
- An SSL certificate with private key in the name of the server hosting the Docker container for Keyfactor Identity Provider to allow administrators to connect to the web-based administration interface for Keyfactor Identity Provider over an encrypted channel.

To prepare the SQL certificate:

1. On the Microsoft SQL server, open the certificates MMC for the local machine store using one of these methods:
    - Using the GUI:
        a. Open an empty instance of the Microsoft Management Console (MMC).
        b. Choose **File->Add/Remove Snap-in...**.
        c. In the *Available snap-ins* column, highlight **Certificates** and click **Add**.
        d. In the Certificates snap-in popup, choose the radio button for Computer account, click **Next**, accept the default of Local computer, and click **Finish**.
        e. Click **OK** to close the Add or Remove Snap-ins dialog.
    - Using the command line:
        a. Open a command prompt using the “Run as administrator” option.
        b. Within the command prompt type the following to open the certificates MMC:

            ```
            certlm.msc
            ```

2. Drill down to the Personal folder under Certificates for the Local Computer. Locate the certificate used to secure connections to your SQL server (see Using SSL to Connect to SQL Server on page 2746).
3. Right-click the certificate and choose All Tasks->Export....

4. Follow the export wizard, choosing not to export the private key, choosing a format of P7B, and opting to including the chain certificates.

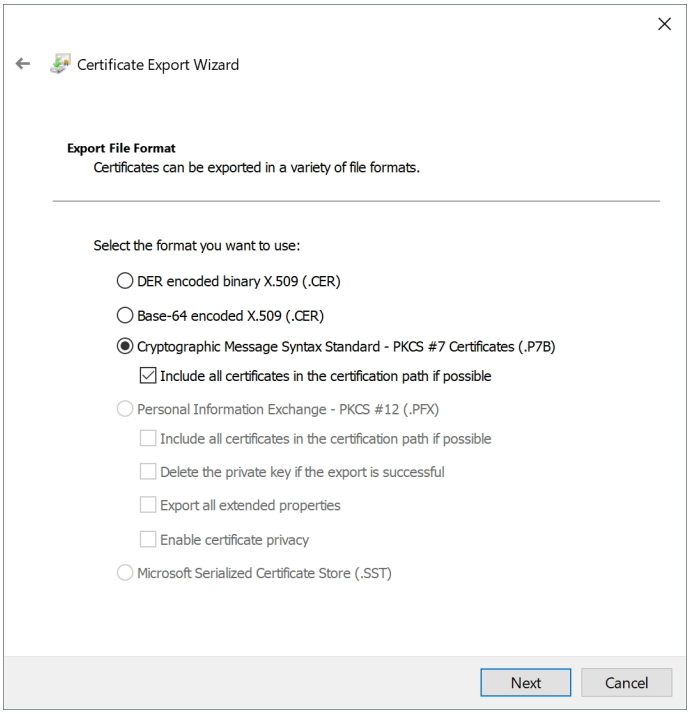


Figure 445: Export the SQL Server Certificate as a P7B

5. Copy the exported file to a working directory on the Docker host.
6. On the Docker host, use OpenSSL to extract the certificate and chain from the P7B file with a command similar to the following:

    ```
    sudo openssl pkcs7 -inform der -in /my/path/mycert.p7b -print_certs -out
    /my/path/mycert.cer
    ```

7. Use the Java keytool command to create a Java Keystore containing all of the certificates that are part of the chain required to trust the certificate used to secure connections to your SQL server (these should be in your incoming .cer file):

    ```
    sudo keytool -import -file /my/path/mycert.cer -keystore /my/path/sql-keystore -
    storepass "MySuperSecureStorePassword"
    ```

To prepare the SSL certificate:

1. Acquire the SSL certificate using the Fully Qualified Domain Name (FQDN) of the server or alias used for the Keyfactor Identity Provider Docker host. This is the name that you will use to access Keyfactor Identity Provider via a browser for management purposes and that Keyfactor

Command will use to access Keyfactor Identity Provider for authentication purposes.

2. Copy the certificate together with its private key to a working directory on the Docker host.

3. Depending on the method you used to acquire your certificate, you may need to manipulate it on the Docker host to get it into the correct format. You need separate PEM-encoded unencrypted private key and certificate files. If your certificate is a PKCS#12 file, you can use OpenSSL commands similar to the following to extract the certificate and key:

   Extract just the certificate, not any chain certificates or the key:

   ```
   sudo openssl pkcs12 -in /my/path/mycert.pfx -clcerts -nokeys -out
   /my/path/mycert.cer
   ```

   Extract just the key:

   ```
   sudo openssl pkcs12 -in /my/path/mycert.pfx -nocerts -out /my/path/mycert_key.pem
   ```

   Decrypt the key:

   ```
   sudo openssl rsa -in /my/path/mycert_key.pem -out /my/path/mycert_key-plain.pem
   ```

   Important: The decrypted key file should be handled carefully and stored securely. During the container deployment, the certificate and key files will be copied to:

   ```
   /install/path/certificates/ssl
   ```

   Permissions should be set on the key file in this location such that the service account running the Docker container has read permission on it and no other users have access of any kind. By default, Docker containers run as root, so the permissions would look like something like this:

   -r-------- 1 root root 1704 Jul 10 09:12 appsrvr18keyexamplecom-key-plain.pem

   Following the deployment, the key file in the working directory (not the directory listed above) should be removed.

SQL Setup

Keyfactor Identity Provider uses Microsoft SQL Server with SQL authentication, not Windows authentication. Your SQL server must be configured to support mixed mode authentication in order to use the SQL authentication. The database for Keyfactor Identity Provider needs to be created in SQL before the deployment is done and appropriate permissions granted for the SQL user you will configure in Keyfactor Identity Provider to make the connection to SQL.

On your Microsoft SQL server:

1. Identify an existing SQL login using SQL authentication (not Windows authentication) or create a new login to be used for Keyfactor Identity Provider to authenticate to SQL.

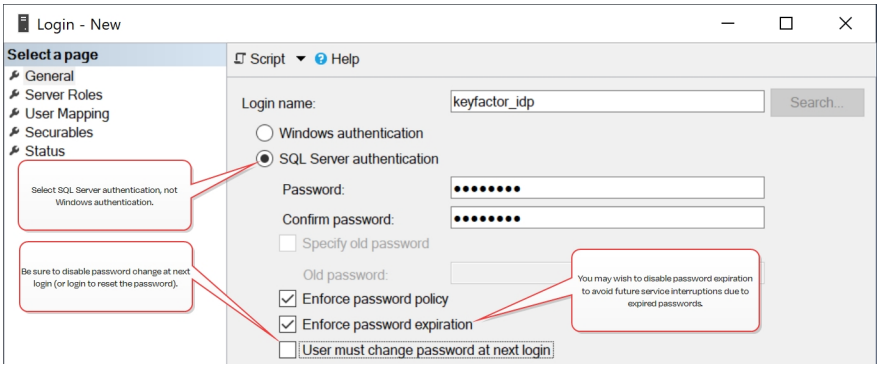


Figure 446: Add a SQL Authentication Login

2. Create a new database in SQL and grant the SQL login you created in the previous step at least dbo permissions on the database. You can do this either by setting it as the database owner while creating the database or by going back into the login after the database is created and granting the access on the User Mapping tab.

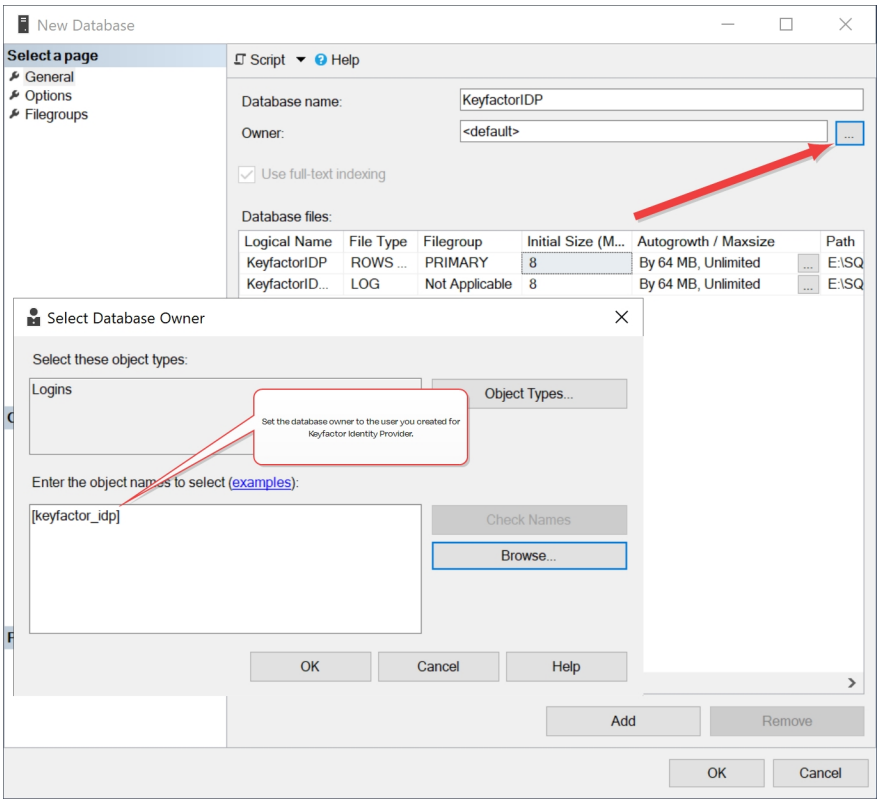


Figure 447: Add a SQL Database

Gather Information

You will need the following information in order to appropriately configure the installation file for the Keyfactor Identity Provider container:

- A username and password for the initial administrative user that will be created in Keyfactor Identity Provider. By default, the username *admin* is used.
- The fully qualified domain name (FQDN) that you will use to access Keyfactor Identity Provider from a browser. This is typically the hostname of the container host.
- The FQDN or IP address of the SQL server hosting the database for Keyfactor Identity Provider. If you choose to use the IP address, the SSL certificate on the SQL server will need to include the IP address as a SAN. If you configured your database in a non-default instance or your SQL is running on a non-standard port, you will need this information as well.
- The name of the database you created in SQL for Keyfactor Identity Provider.
- The username and password for the login you created in SQL for Keyfactor Identity Provider.
- The path to the *sql-keystore* Java keystore you created as per Prepare Certificates on page 2706.
- The paths to the server certificate (*mycert.cer*) and unencrypted private key (*mycert_key-plain.pem*) you prepared as per Prepare Certificates on page 2706.
- The IP address of at least one DNS server in your environment that can be used to resolve the hostname of the SQL server if that server isn't publicly routable.
- The path for the Keyfactor Identity Provider image you will install. The Keyfactor Identity Provider artifactory can be found here:

  keyfactor.jfrog.io/con-develop-us-engineering/command/authentication-server:latest

  Check with your Keyfactor Client Success Manager for credentials.

Installing Using Docker Compose

The following section covers installing Keyfactor Identity Provider using a Docker compose file.

To install Keyfactor Identity Provider in a Linux container and start the container using Docker compose:

1. Create a directory from which you will run the Docker container (e.g. /opt/kyfidp).
2. Copy the *sql-keystore* Java keystore you created (see Prepare Certificates on page 2706) into the directory you created for the Docker container and set the permissions appropriately. It needs to be readable by the user the Docker container will run as (by default root) and its group. For example:

   ```
   sudo chown root:root sql-keystore
   ```

```
sudo chmod 440 sql-keystore
```

3. Copy the certificate and key for the Docker host (see Prepare Certificates on page 2706) into the directory you created for the Docker container and set the permissions appropriately. They need to be readable by the user the Docker container will run as (by default root) and its group. For example:

```
sudo chown root:root appsrvr18keyexamplecom-server.cer

sudo chown root:root appsrvr18keyexamplecom-key-plain.pem

sudo chmod 440 appsrvr18keyexamplecom-server.cer

sudo chmod 440 appsrvr18keyexamplecom-key-plain.pem
```

4. From your Docker host, retrieve the Keyfactor Identity Provider image from the artifactory with commands similar to the following (using credentials provided to you by Keyfactor; the password is saved in my_password.txt):

```
cat my_password.txt | sudo docker login keyfactor.jfrog.io --username username --
password-stdin

sudo docker pull keyfactor.jfrog.io/con-develop-us-
engineering/command/authentication-server:latest
```

Important: Remove the my_password.txt file when complete.

5. Create a Docker compose file (*compose.yaml*) in the directory you created for the Docker container similar to the following, using inputs as per Table 844: Keyfactor Identity Provider Container Parameters and referencing the artifactory you pulled. The fields highlighted in red below indicate fields that need to be edited or that you may wish to edit.

Important: When editing the file, be sure to preserve the indenting exactly as found. YAML requires a very specific file layout to function. If the indenting (multiples of two spaces) or layout is incorrect, you will receive an error when trying to install.

```
services:
  auth:
    image: keyfactor.jfrog.io/con-develop-us-engineering/command/authentication-server:latest
    container_name: kyfidp
    ports:
```

```
      - "1443:8443"
    environment:
      KC_HTTPS_CERTIFICATE_FILE: /etc/x509/https/tls.crt
      KC_HTTPS_CERTIFICATE_KEY_FILE: /etc/x509/https/tls.key
      KC_SPI_THEME_DEFAULT: Keyfactor-Keycloak-Theme

      KEYCLOAK_ADMIN: admin
      KEYCLOAK_ADMIN_PASSWORD: 'MySuperSecureAdminPassword' # The password needs quotes under
some circumstances if it contains special characters

      KC_HOSTNAME: appsrvr18.keyexample.com

      # This field is only required if you're using a port other than 443
      KC_HOSTNAME_PORT: 1443

      # This user must be dbo on KC_DB_URL_DATABASE
      KC_DB_USERNAME: keyfactor_idp
      KC_DB_PASSWORD: 'MySuperSecureSQLUserPassword' # The password needs quotes under some
circumstances if it contains special characters
      KC_DB_URL_HOST: sqlsrvr05.keyexample.com
      KC_DB_URL_DATABASE: KeyfactorIDP
      KC_DB: mssql
      KC_TRANSACTION_XA_ENABLED: false
      KC_DB_URL_PROPERTIES: ';encrypt-
t=true;trustServerCertificate=false;sendStringParametersAsUnicode=false;Integrated Secur-
ity=False;Persist Security Info=True;trustStore=/temp/sql-
keystore;trustStorePassword=MySuperSecureJKSStorePassword;'

      # This value must be configured even if you do not have a reverse proxy
      KC_PROXY: none

    command:
      - start --import-realm

    volumes:
      - ./appsrvr18keyexamplecom-server.cer:/etc/x509/https/tls.crt
      - ./appsrvr18keyexamplecom-key-plain.pem:/etc/x509/https/tls.key
      - ./sql-keystore:/temp/sql-keystore

    # Optionally set the DNS server(s) for the Keyfactor Identity Provider server
    dns:
      - 192.168.12.2
      - 192.168.12.3
    restart: always
```

6. Set the permissions on the *compose.yaml* file such that the file is owned by root and readable only by root (this assumes your Docker daemon is running as root, which is typical). For example:

```
sudo chown root:root compose.yaml

sudo chmod 400 compose.yaml
```

> **Tip:** If you need to make edits to the compose file, you will need to make the file writable again. For example:
>
> ```
> sudo chmod 600 compose.yaml
> ```

7. Execute the following command to install and run the container in the foreground:

```
sudo docker compose up
```

You can instead run it in the background by adding the -d flag like so, but it can sometimes be helpful to run it in the foreground initially so that you can easily review the log output live:

```
sudo docker compose up -d
```

> **Tip:** To stop and start the container again after installation is complete, use the following commands:
>
> ```
> sudo docker compose stop
>
> sudo docker compose start
> ```
>
> Or:
>
> ```
> sudo docker compose restart
> ```
>
> If you need to delete the container and try the install again, use this command:
>
> ```
> sudo docker compose down
> ```
>
> This will not remove the configurations made in the SQL database.
>
> To review logs generated from the container, identify the container ID or name with this command:
>
> ```
> sudo docker container ls
> ```
>
> Then use the following command to output the current log (with the optional --follow to make output continuous):

```
sudo docker container logs [--follow] [container ID or name]
```

Table 844: Keyfactor Identity Provider Container Parameters

Section	Parameter	Description
image		**Required***. The path to the artifactory and image for the Keyfactor Identity Provider implementation.
container_name		A name to give to the container, if desired, for ease of reference.
environment	KC_HTTPS_CERTIFICATE_FILE	**Required**. The path and filename of the location within the container where the SSL certificate for the Docker host (see Prepare Certificates on page 2706) will live.
environment	KC_HTTPS_CERTIFICATE_KEY_FILE	**Required**. The path and filename of the location within the container where the SSL certificate key for the Docker host (see Prepare Certificates on page 2706) will live.
environment	KC_SPI_THEME_DEFAULT	**Required**. The theme for the Keyfactor Identity Provider implementation.
environment	KEYCLOAK_ADMIN	**Required**. The username for the initial administrative user for Keyfactor Identity Provider. The default is *admin*.
environment	KEYCLOAK_ADMIN_PASSWORD	**Required***. Set a secure password for the initial administrative user for Keyfactor Identity Provider.
environment	KC_HOSTNAME	**Required***. The is the fully qualified domain name of the Docker host where you are deploying your container.
environment	KC_HOSTNAME_PORT	The port number you will use to access Keyfactor Identity Provider via a browser. This field only needs to be populated if you won't be using 443. If you'll be using 443, the entry should be commented out or removed.
environment	KC_DB	**Required***. The type of SQL server. Only Microsoft SQL Server is supported (mssql).
environment	KC_DB_URL_HOST	**Required***. The fully qualified domain name of the Microsoft SQL server that will host the database for Keyfactor Identity Provider.
environment	KC_DB_URL_DATABASE	**Required***. The name of the Keyfactor Identity Provider database you pre-created in SQL per SQL Setup on page 2708.

Section	Parameter	Description
environment	KC_DB_ PASSWORD	**Required***. The password for the SQL login to which you granted database ownership permissions on the Keyfactor Identity Provider database per SQL Setup on page 2708.
environment	KC_DB_ USERNAME	**Required***. The username for the SQL login to which you granted database ownership permissions on the Keyfactor Identity Provider database per SQL Setup on page 2708.
environment	KC_DB_URL_ PROPERTIES	**Required***. The SQL database connection string including the password you set to secure the Java keystore that holds the SQL server's certificate as per Prepare Certificates on page 2706.
environment	KC_ TRANSACTION_ XA_ENABLED	A Boolean that indicates whether the database for the installation supports XA transactions.
environment	KC_PROXY	The proxy address forwarding mode if the server is behind a reverse proxy.
ports		The first number in the ports field indicates the port number you will use to access Keyfactor Identity Provider via a browser and that Keyfactor Command will use to access Keyfactor Identity Provider. If there are no other containers using this port on this Docker host, you may use 443. If 443 is already in use, you will need to change this to an alternate port (e.g. 1443). The second number in the ports field indicates the port number the Docker container uses internally. Do not change this number.
command		The command to start the container and import the realm JSON.
volumes		In this section you set the file names for the certificate and key files you copied into the directory for your Docker container. The volumes section sets mappings between files that exist on the host and locations in the running container.
dns		In this section, add at least one IP for a DNS server that can be used to resolve hostname information for your SQL server from the Keyfactor Identity Provider container if the SQL server's address is not externally routable.

Once the Keyfactor Identity Provider completes successfully, you should be able to open the administration console for it in a browser and gather the information you will need to complete the Keyfactor Command installation referencing it.

1. Use a browser to open the Keyfactor Identity Provider management interface. For example:

   ```
   https://appsrvr18.keyexample.com:1443
   ```

   Click the **Administration Console** link and sign in with the initial administrative user and password you defined with the KEYCLOAK_ADMIN and KEYCLOAK_ADMIN_PASSWORD settings.

   Note: Keyfactor Command communicates with Keyfactor Identity Provider over HTTPS, so be sure that you are working with Keyfactor Identity Provider over HTTPS to confirm that it is working correctly with no certificate errors.

2. In the Keyfactor Identity Provider Administration Console, select Keyfactor in the realm drop-down.

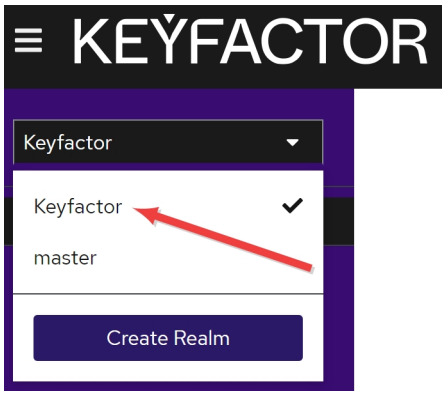
   

   Figure 448: Select a Realm in the Keyfactor Identity Provider Administration Console

3. In the Keyfactor Identity Provider Administration Console, browse to *Clients* > *Client list* and click the *Command-OIDC-Client* client.

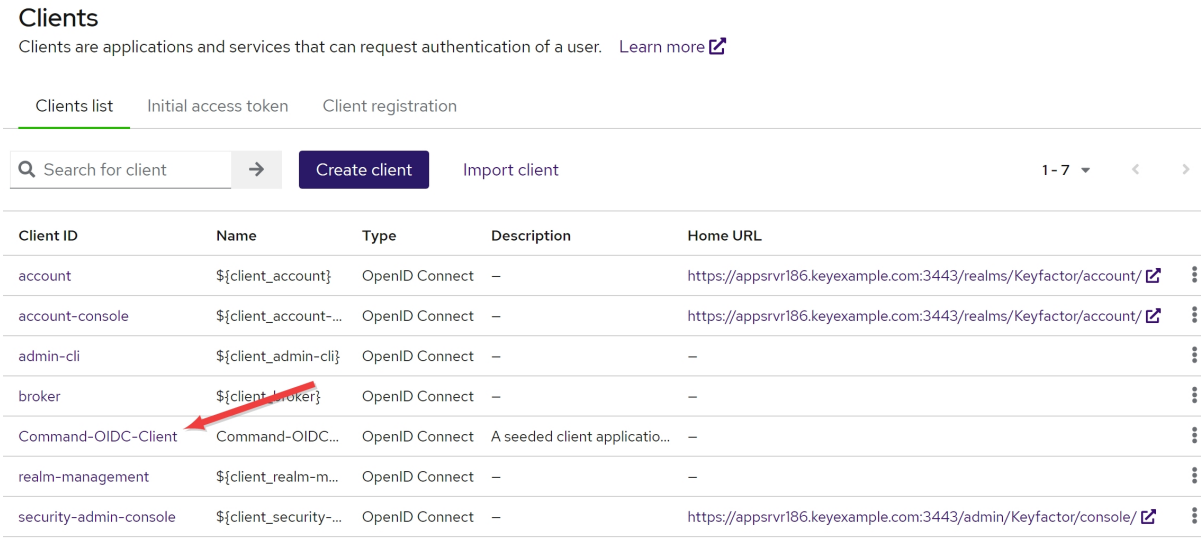


Figure 449: Select Command-OIDC-Client in the Keyfactor Identity Provider Administration Console

4. In the Client details, select the Credentials tab and click **Regenerate** next to the *Client secret* field. When prompted, answer Yes to regenerate the secret.

> **Important:** This step is necessary to set a unique, complex secret for your environment. **Do not skip this step**.

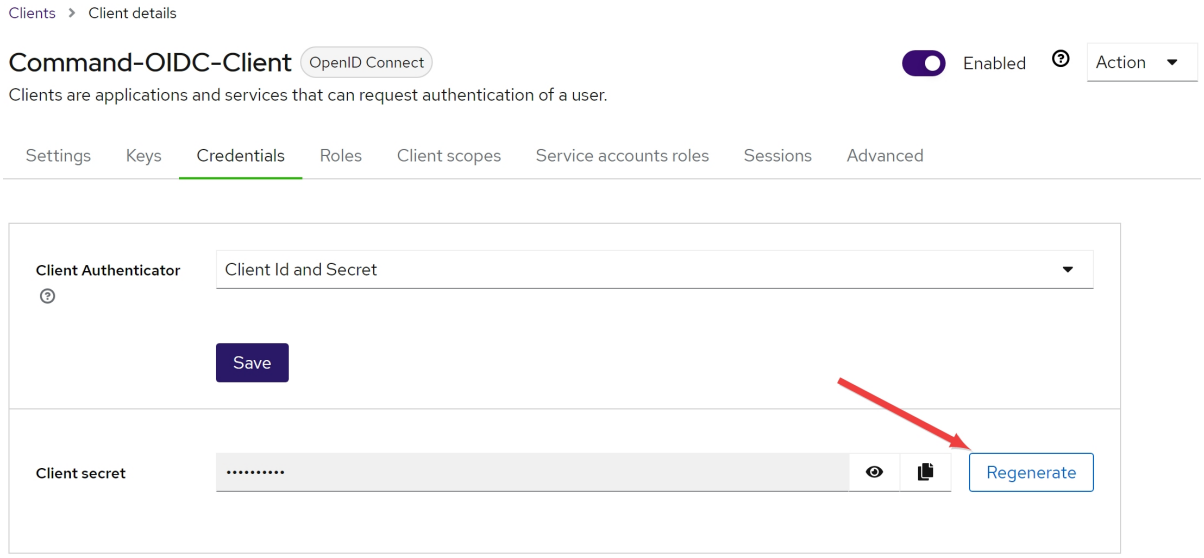


Figure 450: Regenerate the Keyfactor Identity Provider Secret

5. In the Client details on the Credentials tab click the **Copy** button next to the *Client secret* field to copy the unmasked version of the client secret to the clipboard (you do not need to display it

unmasked first) and save this in a secure location. You will need it during the Keyfactor Command configuration.

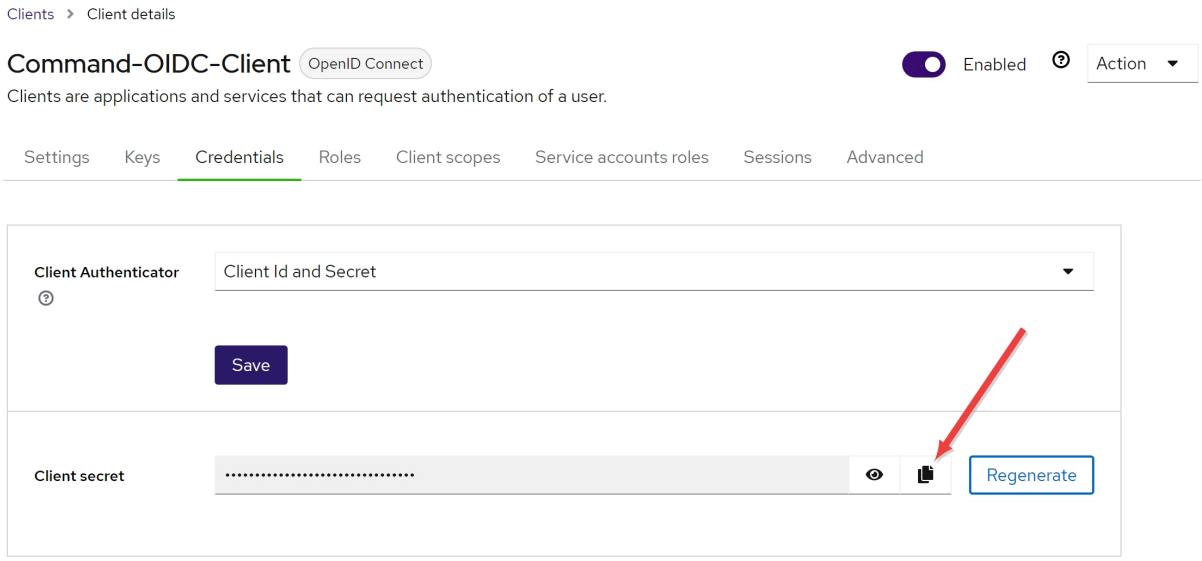


Figure 451: Copy the Keyfactor Identity Provider Secret

6. In the Client details on the Settings tab, populate the *Valid redirect URIs* and *Valid post logout redirect URIs* fields. The values for these fields are made up of the fully qualified domain name or alias you will use to access your Keyfactor Command server, the virtual directory name you will use to access the Keyfactor Command Management Portal (KeyfactorPortal by default), a specific endpoint for the URI, and the name you will give to Keyfactor Identity Provider when configuring Keyfactor Command (see Authentication Tab on page 2787). For example:

    - Valid redirect URIs:

      ```
      https://keyfactor.kexample.com/KeyfactorPortal/callback/Command-OIDC
      ```

    - Valid post logout redirect URIs:

      ```
      https://keyfactor.kexample.com/KeyfactorPortal/signout-callback/Command-OIDC
      ```

    Important: Case matters for the virtual directory name—use *KeyfactorPortal* rather than *keyfactorportal* if you plan to accept the default virtual directory name.

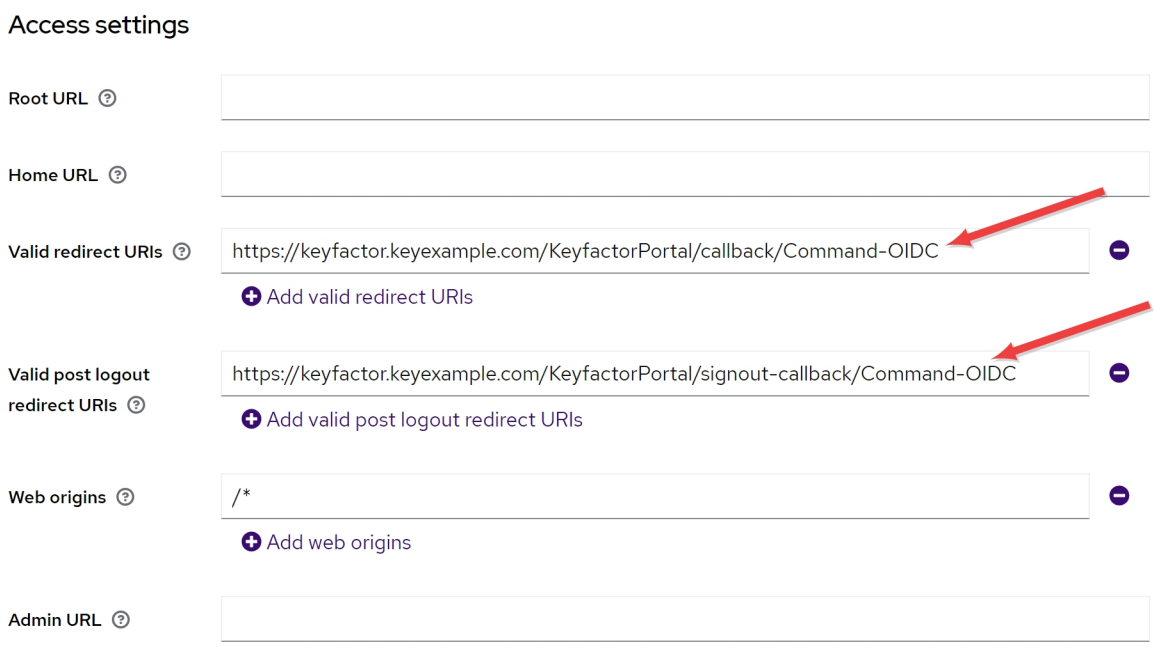


Figure 452: Set the Client Access Settings

7. In the Keyfactor Identity Provider Administration Console, browse to *Realm settings* and select the General tab. On the General tab, click the **OpenID Endpoint Configuration** link. This will open in a new browser window.

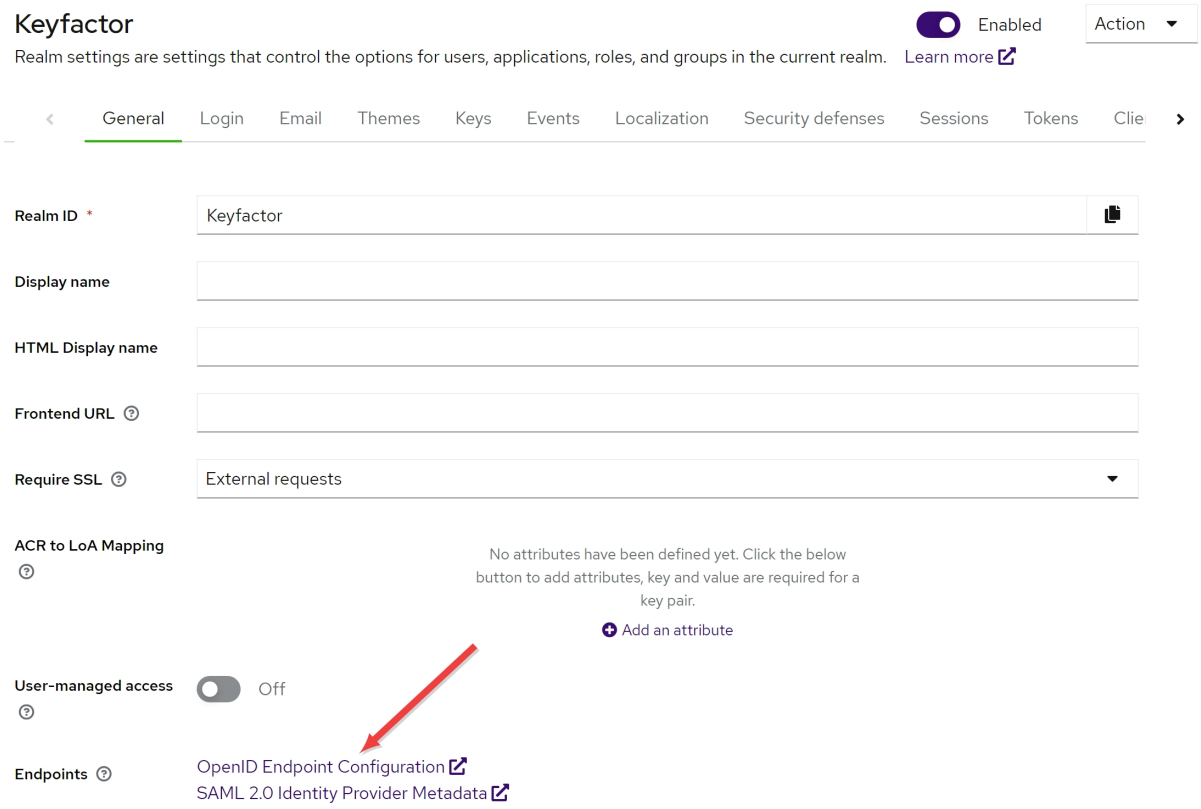


Figure 453: OpenID Endpoint Configuration Link

8. In the browser window for the **OpenID Endpoint Configuration** link, review the settings. You may find it helpful to use a JSON formatting browser extension to make the data easier to read. The data you need from this configuration info is:

    - Issuer (a.k.a. Authority)
    - Authorization Endpoint
    - Token Endpoint
    - User Info Endpoint
    - jwks_uri (a.k.a. JSONWebKeySetUri)

    Make note of these URLs, without the quotation marks. You will need them during the Keyfactor Command configuration.

```
object {53}
  issuer : "https://appsrvr186.keyexample.com:3443/realms/Keyfactor"
  authorization_endpoint : "https://appsrvr186.keyexample.com:3443/realms/Keyfactor/protocol/openid-connect/auth"
  token_endpoint : "https://appsrvr186.keyexample.com:3443/realms/Keyfactor/protocol/openid-connect/token"
  introspection_endpoint : "https://appsrvr186.keyexample.com:3443/realms/Keyfactor/protocol/openid-connect/token/introspect"
  userinfo_endpoint : "https://appsrvr186.keyexample.com:3443/realms/Keyfactor/protocol/openid-connect/userinfo"
  end_session_endpoint : "https://appsrvr186.keyexample.com:3443/realms/Keyfactor/protocol/openid-connect/logout"
  frontchannel_logout_session_supported : true
  frontchannel_logout_supported : true
  jwks_uri : "https://appsrvr186.keyexample.com:3443/realms/Keyfactor/protocol/openid-connect/certs"
  check_session_iframe : "https://appsrvr186.keyexample.com:3443/realms/Keyfactor/protocol/openid-connect/login-status-iframe.html"
```


Figure 454: OpenID Endpoint Configuration Settings

9. In the Keyfactor Identity Provider Administration Console, browse to *Realm settings* and select the Sessions tab.

   On the Sessions tab, locate the *SSO Session Max* value. This value should match the *Session Expiration* parameter value configured in the Keyfactor Command configuration wizard on the Authentication tab. The *Session Expiration* value determines the length of time a browser session in the Keyfactor Command Management Portal will remain logged in before the user is prompted to re-authenticate regardless of whether the session is idle or in active use.

   Locate the *SSO Session Idle* value and set it to a value that is appropriate to your environment. This value determines the length of time an idle browser session in the Keyfactor Command Management Portal will remain logged in before automatically logging out the user if no input from the user is received.

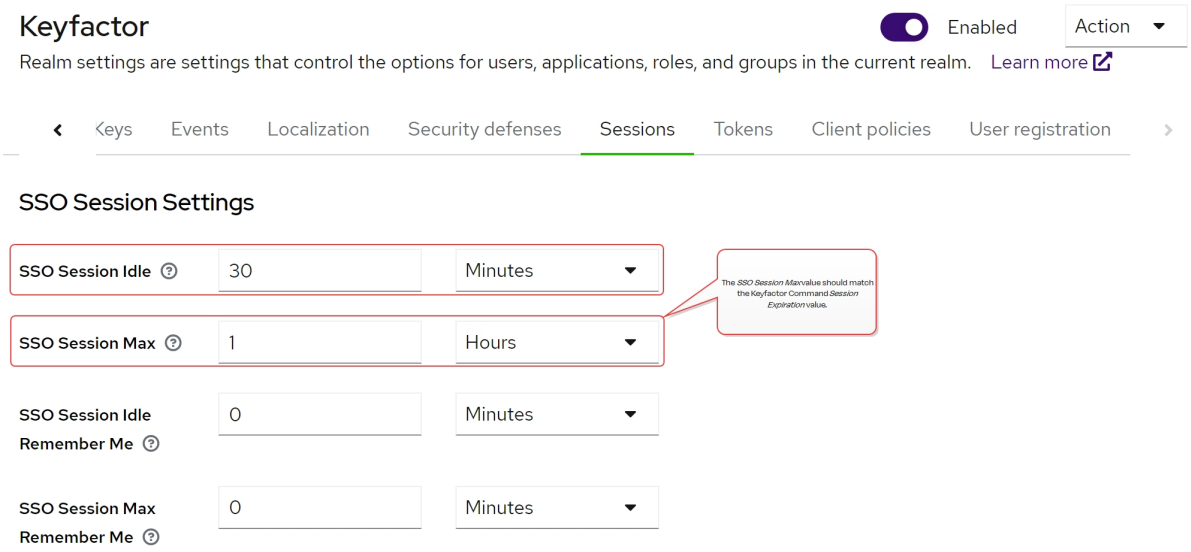


Figure 455: SSO Session Values

10. In the Keyfactor Identity Provider Administration Console, browse to *Realm settings* and select the Tokens tab. On the Tokens tab, locate the *Access Token Lifespan* value. This value should be greater than or equal to the *Cookie Expiration* parameter value configured in the Keyfactor Command configuration wizard on the Authentication tab

The *Cookie Expiration* value determines the length of time the authentication cookie for the Keyfactor Command Management Portal browser session is considered valid. After half of the setting's duration, Keyfactor Command will attempt to use a refresh token to update the cookie. If this fails, the user's session will be terminated. The cookie renewal is seamless from the user's perspective (there is no prompt for credentials).

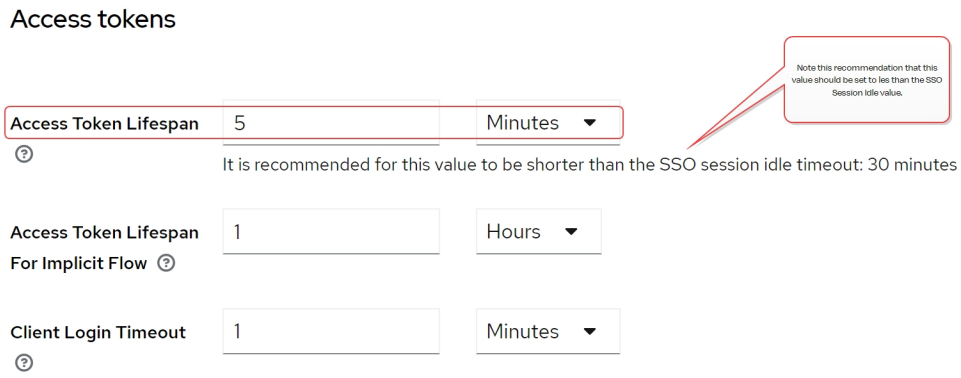


Figure 456: Access Token Lifespan

11. In the Keyfactor Identity Provider Administration Console, browse to *Users*. Click **Add user** to add at least one new user to be granted administrative permissions in Keyfactor Command during the Keyfactor Command installation.

> **Note:** The admin user created during the Keyfactor Identity Provider installation can't be used for Keyfactor Command authentication because it is in the master realm, not the Keyfactor realm.

> **Tip:** Once the Keyfactor Command installation is complete, additional users and groups that you have added into Keyfactor Identity Provider can be added through the Keyfactor Command Management Portal and granted varying roles; only one user is required initially so a user can open the Management Portal at the conclusion of the installation.

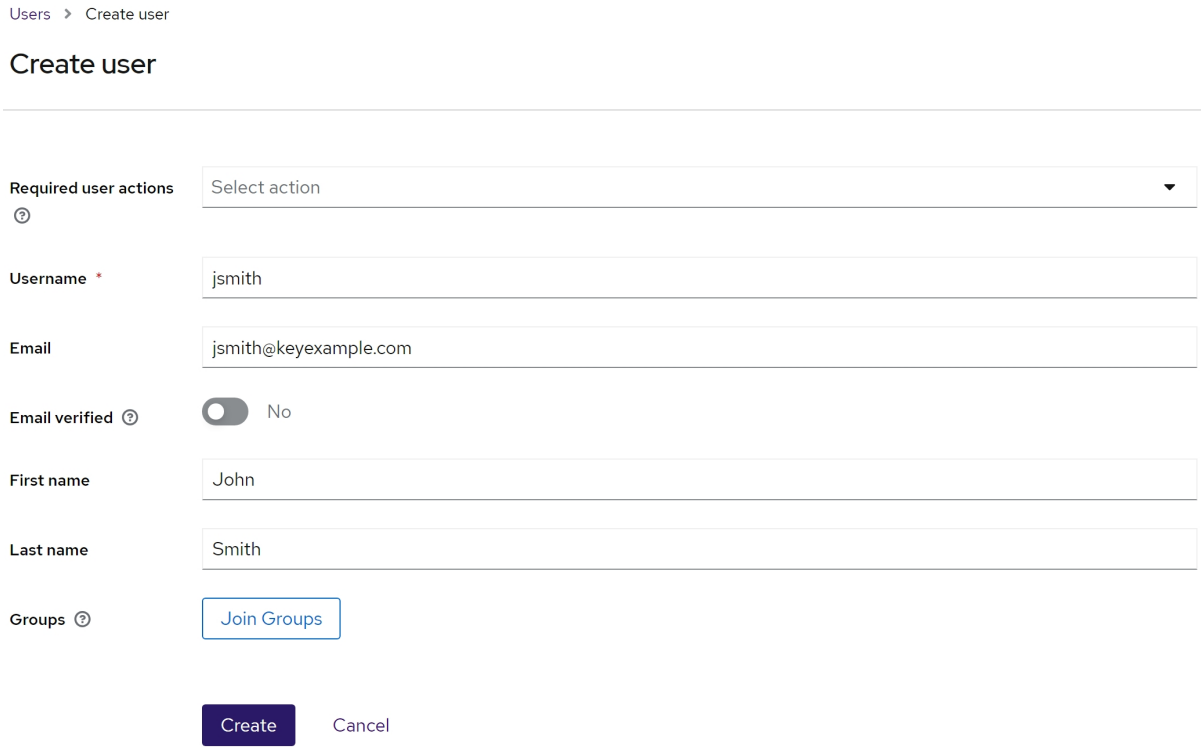


Figure 457: Add a Keyfactor Identity Provider User

12. Once the user account creation is complete, on the user details locate the ID for the user and make a copy of the GUID. This GUID is used to reference the user account when you configure the user as an administrator in the Keyfactor Command configuration wizard.

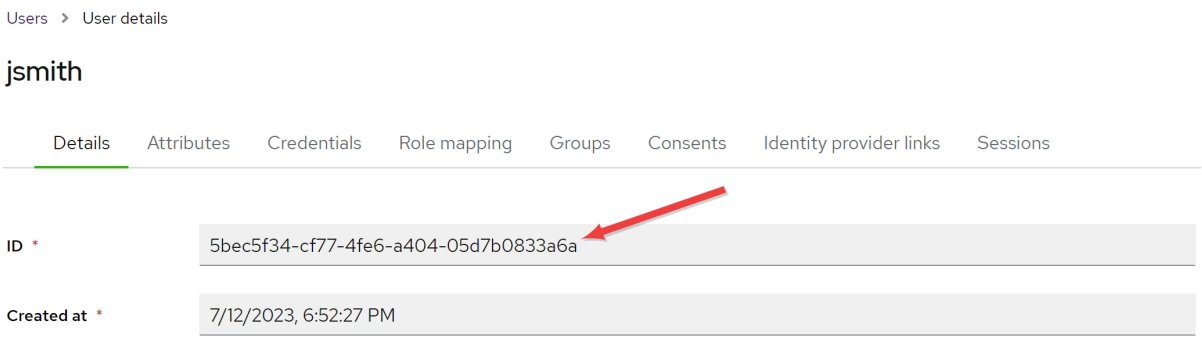


Figure 458: Locate the Keyfactor Identity Provider User's ID

13. In the user details on the Credentials tab, click **Set password** and set a password for the new user.

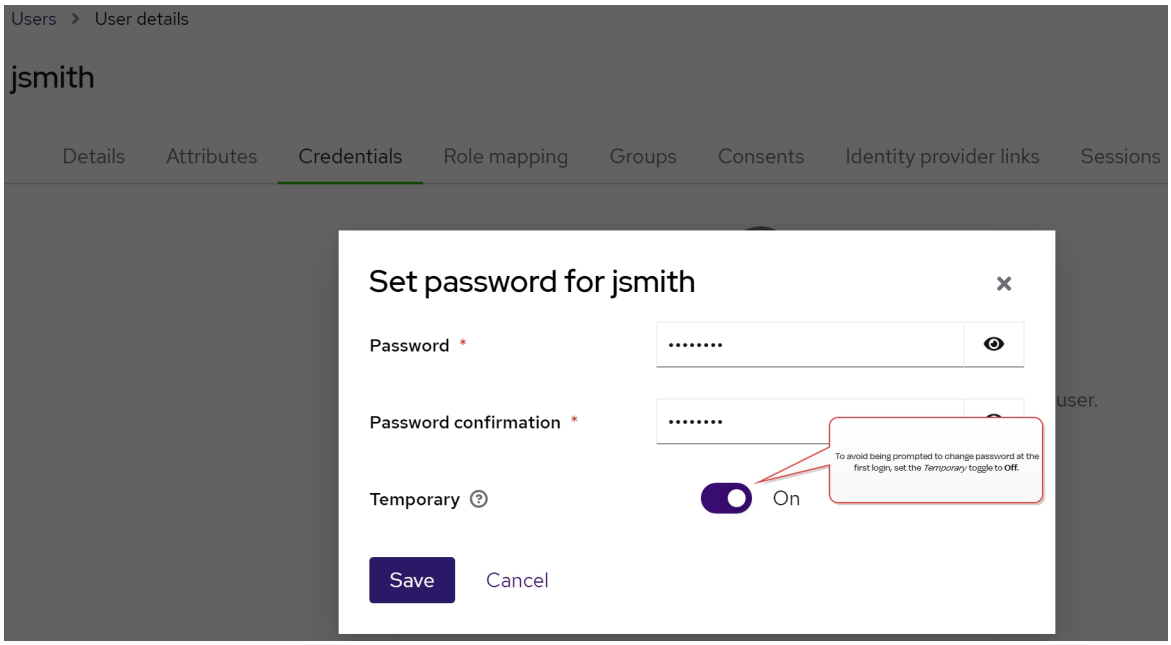


Figure 459: Set a Password for the Keyfactor Identity Provider User

From here you can create more administrative users, standard users, and groups (see Using Keyfactor Identity Provider below) or configure federation to an alternate OAuth provider (see Federating from Keyfactor Identity Provider on page 2732).

Using Keyfactor Identity Provider

Once you have finished configuring Keyfactor Identity Provider, you're ready to add roles, optional groups, users, and service accounts into it to be used for authentication to Keyfactor Command. Alternatively, you may choose to federate to an additional OAuth provider (see Federating from Keyfactor Identity Provider on page 2732), in which case you don't need to add users in Keyfactor Identity Provider, but you will still need roles, optional groups, and service accounts, since it's the roles in Keyfactor Identity Provider that are used to create claims in Keyfactor Command to grant access to users holding these roles.

Note: You can grant access to Keyfactor Command on a user-by-user basis rather than with roles, but the management overhead of this method is much greater. Keyfactor recommends using roles.

Roles and Groups

To add roles and groups in Keyfactor Identity Provider:

1. Use a browser to open the Keyfactor Identity Provider management interface. For example:

   ```
   https://appsrvr18.keyexample.com:1443
   ```

Click the **Administration Console** link and sign in with an administrative user and password (see Installing Using Docker Compose on page 2710).

2. In the Keyfactor Identity Provider Administration Console, select Keyfactor in the realm drop-down.

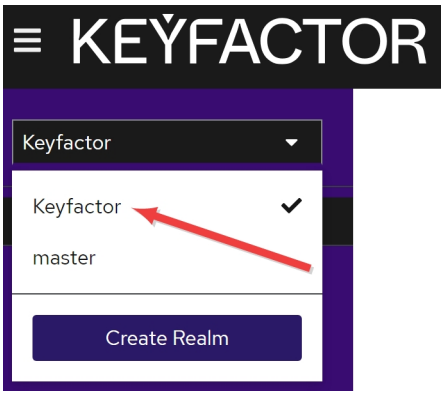


Figure 460: Select a Realm in the Keyfactor Identity Provider Administration Console

3. In the Keyfactor Identity Provider Administration Console, browse to *Realm roles*. Click **Create role** to add a new role to be used to grant permissions in Keyfactor Command. Enter a **Role name** and **Description**.

> **Note:** The Role name is used when referencing the role from Keyfactor Command to create a claim and map it to a security role to grant permissions to users.

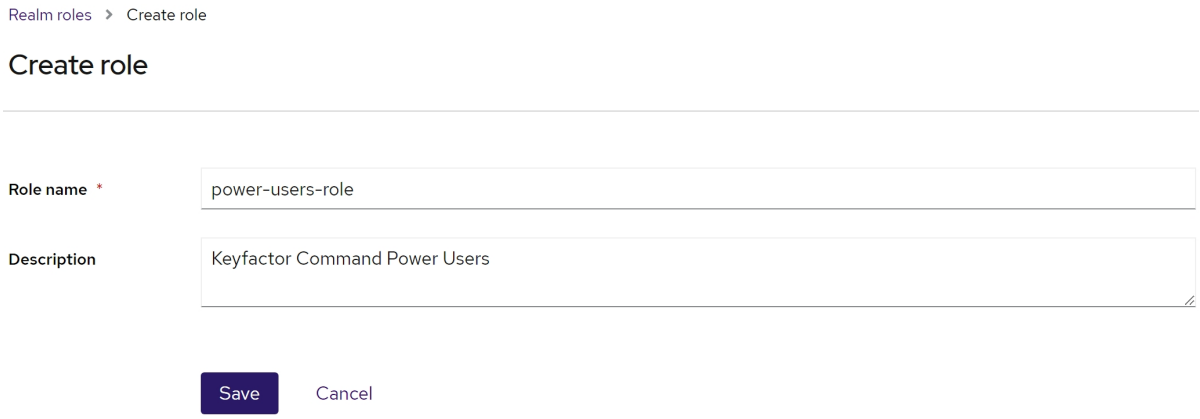


Figure 461: Add a Keyfactor Identity Provider Role

Repeat this step for each role that you will use from Keyfactor Command. For example, administrators, power users, and limited access users.

4. If desired, you can organize your roles into groups. This can simplify the process of assigning the roles to your users. To create a group, in the Keyfactor Identity Provider Administration Console, browse to *Groups*. Click **Create group** to add a new organizational group. Enter a **Name** for the group.

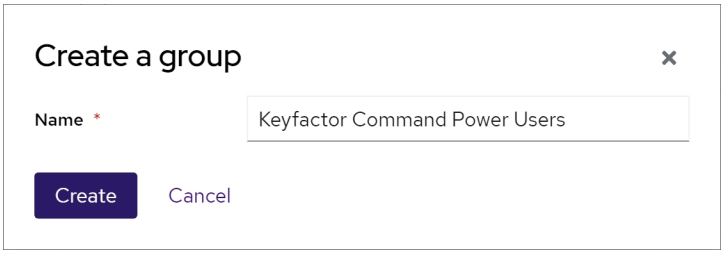


Figure 462: Add a Keyfactor Identity Provider Group

5. Once the group creation is complete, open the group details. In the group details on the Role mapping tab, click **Assign role** and select the role or roles to assign to this group.

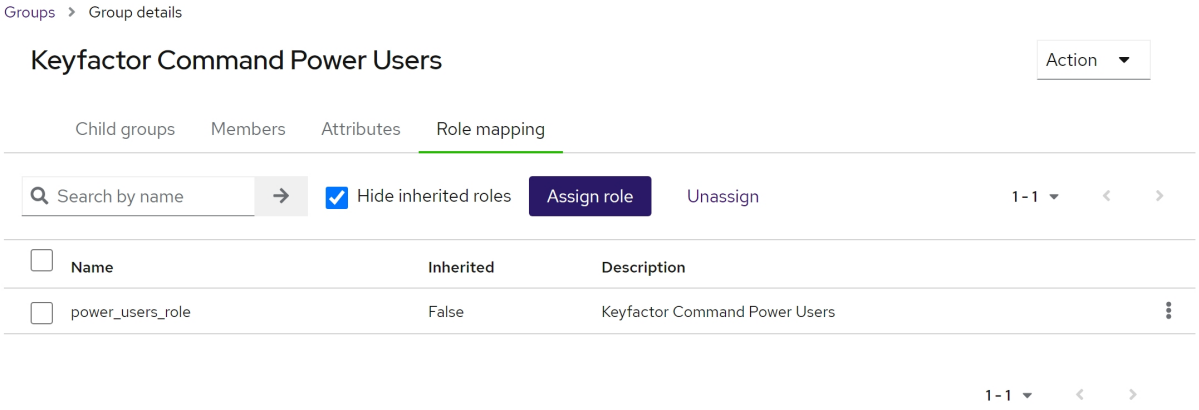


Figure 463: Assign a Role to a Group in Keyfactor Identity Provider

Repeat these two steps for each group that you will use to manage roles in Keyfactor Identity Provider.

Users

Be sure to create your roles and groups before adding your users.

To add users in Keyfactor Identity Provider:

1. Use a browser to open the Keyfactor Identity Provider management interface. For example:

   ```
   https://appsrvr18.keyexample.com:1443
   ```

   Click the **Administration Console** link and sign in with an administrative user and password (see Installing Using Docker Compose on page 2710).

2. In the Keyfactor Identity Provider Administration Console, select Keyfactor in the realm drop-down.

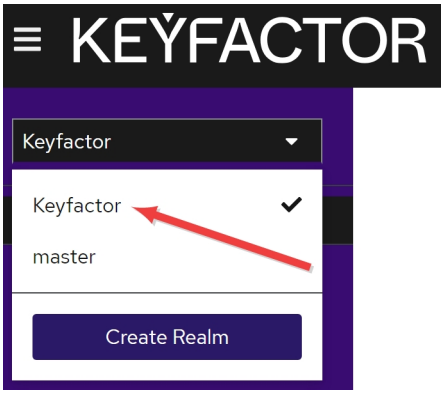


Figure 464: Select a Realm in the Keyfactor Identity Provider Administration Console

3. In the Keyfactor Identity Provider Administration Console, browse to *Users*. Click **Add user** to add a user. Enter at minimum a **Username**, and click **Join Groups**. In the *Select groups to join* dialog, select an appropriate group for this user and click **Join**.

> **Tip:** By joining a group, your user now inherits the roles of this group.

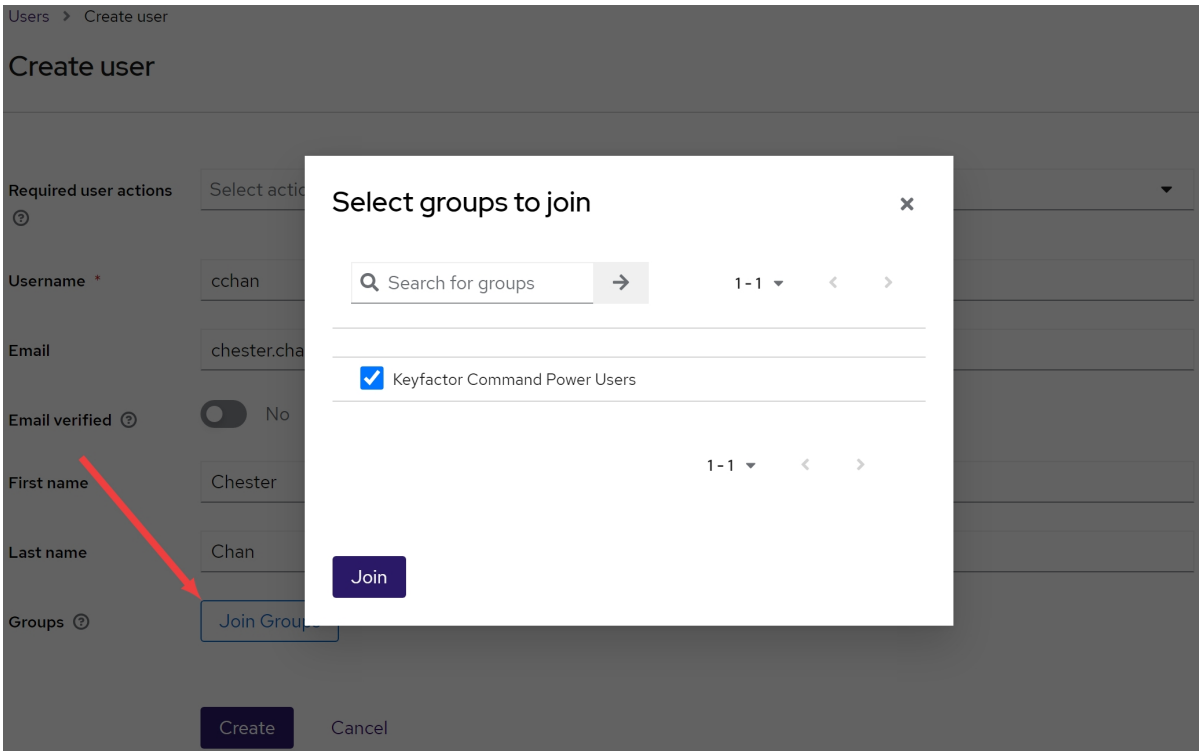


Figure 465: Add a Keyfactor Identity Provider User

4. Once the user creation is complete, open the user details. In the user details on the Credentials tab, click **Set password** and set a temporary password for the new user. The user will be prompted to set a new password on initial logon unless you toggle the **Temporary** option to **Off**.

> **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

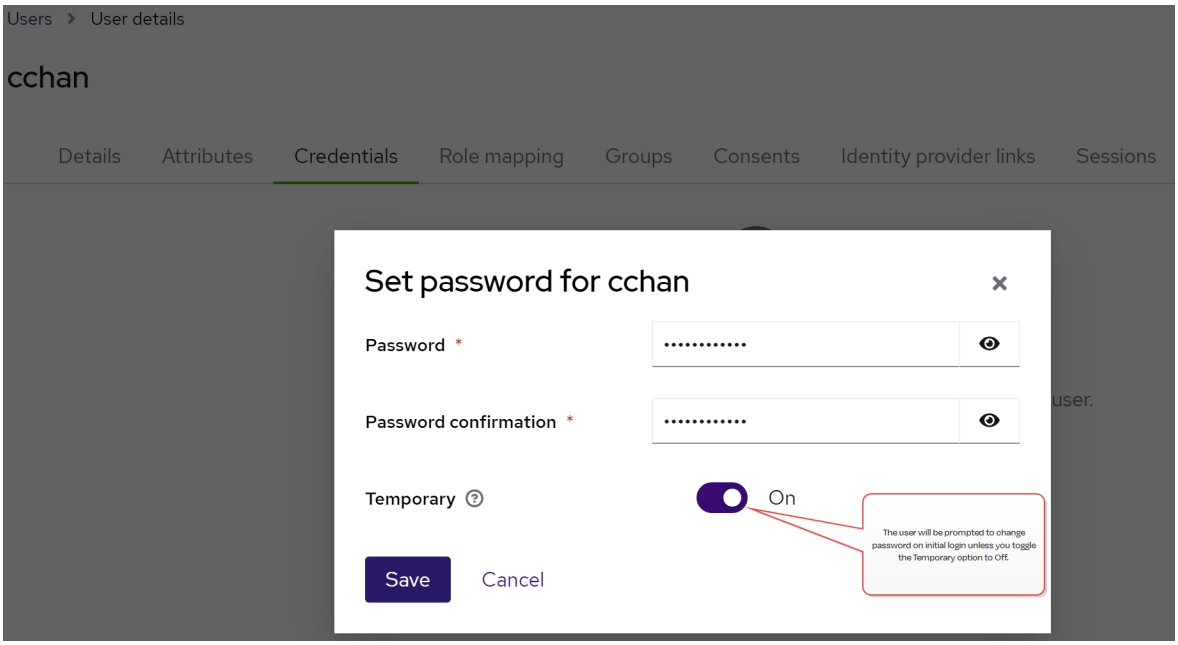


Figure 466: Set a Password for the Keyfactor Identity Provider User

Repeat these steps for each user who will access Keyfactor Command using an identity provider other than Active Directory.

5. If you prefer to add roles directly rather than via groups, in the user details on the Role mapping tab, click **Assign role** and select a role for the new user.

> **Tip:** Roles assigned via group membership won't appear on the Role mapping tab unless you uncheck the **Hide inherited roles** checkbox.

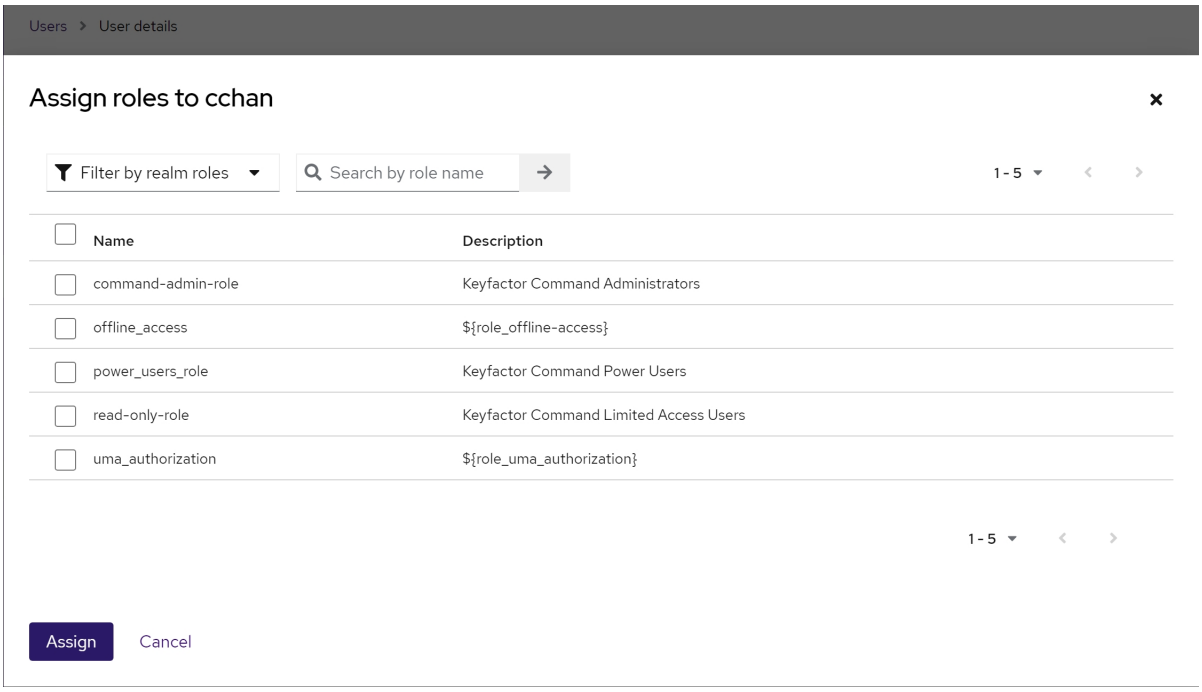


Figure 467: Assign a Role to a Keyfactor Identity Provider User

Service Accounts

Keyfactor Command uses client records in Keyfactor Identity Provider to provide some service account functions. You will or may need this type of service account if you plan to:

- Install Keyfactor Command (a service account is added to allow Keyfactor Command to make API requests to Keyfactor Identity Provider)
- Use the Keyfactor API
- Use the Keyfactor Universal Orchestrator

Note: Client accounts for these functions should be created in Keyfactor Identity Provider even if you plan to use federation for your users. Token authentication requests for these type of functions are not federated.

To add clients for service account in Keyfactor Identity Provider:

1. Use a browser to open the Keyfactor Identity Provider management interface. For example:

   ```
   https://appsrvr18.keyexample.com:1443
   ```

   Click the **Administration Console** link and sign in with an administrative user and password (see Installing Using Docker Compose on page 2710).

2. In the Keyfactor Identity Provider Administration Console, select Keyfactor in the realm drop-down.

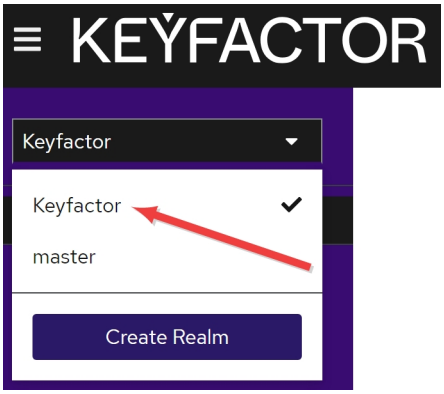


Figure 468: Select a Realm in the Keyfactor Identity Provider Administration Console

3. In the Keyfactor Identity Provider Administration Console, browse to *Clients*. Click **Create client** to add a service account. On the General Settings tab, select a **Client type** of *OpenIdConnect* and enter a unique **Client ID**. This Client ID will be how you will reference the service account from Keyfactor Command. It should not contain spaces. Give the client a **Name** and **Description**. Toggle the **Always display in UI** option to *On* to allow the account to always appear in the UI even when it's not in active use. Click **Next**.

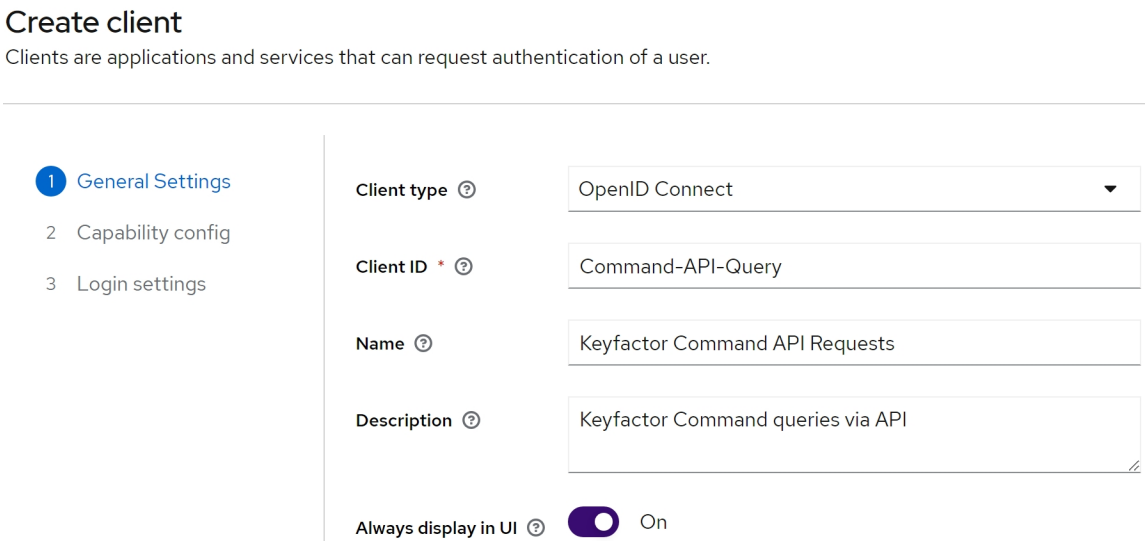


Figure 469: Add a Keyfactor Identity Provider Service Account (Client): General

4. On the Capability config tab, toggle **Client authentication** to *On* and in the Authentication flow section, uncheck everything except **Service accounts roles**. Click **Next**.

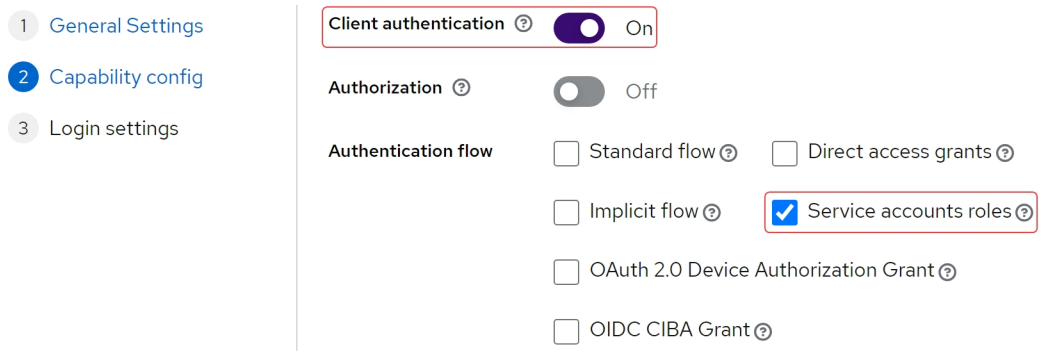


Figure 470: Add a Keyfactor Identity Provider Service Account (Client): Capabilities

5. On the Login settings tab, click **Save**. You do not need to populate any of the data on this tab.

6. In the Client details on the Credentials tab, click the **Copy** button next to the *Client secret* field to copy the unmasked version of the client secret to the clipboard (you do not need to display it unmasked first) and save this in a secure location. For the service account Keyfactor Command uses to make API requests, you will need this and the Client ID during the Keyfactor Command configuration.

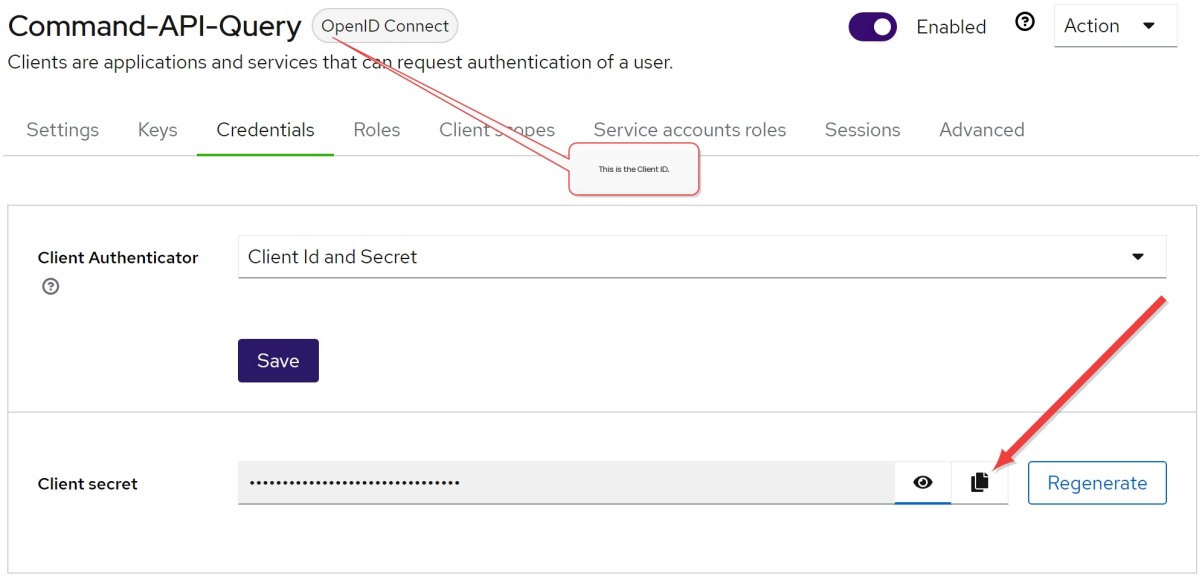


Figure 471: Copy the Keyfactor Identity Provider Service Account (Client) Secret

Federating from Keyfactor Identity Provider

Keyfactor Command can be used with a variety of OAuth 2.0 compliant providers via federation through Keyfactor Identity Provider. Once you have finished configuring Keyfactor Identity Provider, you're ready to federate to an additional OAuth provider, if desired. You may choose to manage your users and groups in Keyfactor Identity Provider and not add federation to an additional provider. Federation can be added at any time. The examples below show configuring Okta as a federated

identity provider. If you're not using Okta, this may give you sufficient information to configure the identity provider of your choice, since configuration tends to be similar.

The below configuration steps assume that you have already completed the installation of Keyfactor Identity Provider (see Installing Keyfactor Identity Provider on page 2705) and have created at least one role in Keyfactor Identity Provider that you will use to grant permissions in Keyfactor Command (see Using Keyfactor Identity Provider on page 2724). Keyfactor Command access control with Keyfactor Identity Provider and federation works by granting permissions in Keyfactor Command to Keyfactor Identity Provider roles, assigning those roles to users in Keyfactor Identity Provider, and importing user records from the federated identity provider automatically on login for any user attempting to access Keyfactor Command (see Figure 473: Federated Identity Provider Login Flow). The user accounts in the federated identity provider need to be assigned roles that mirror the roles in Keyfactor Identity Provider to provide a seamless first-time login experience for users.

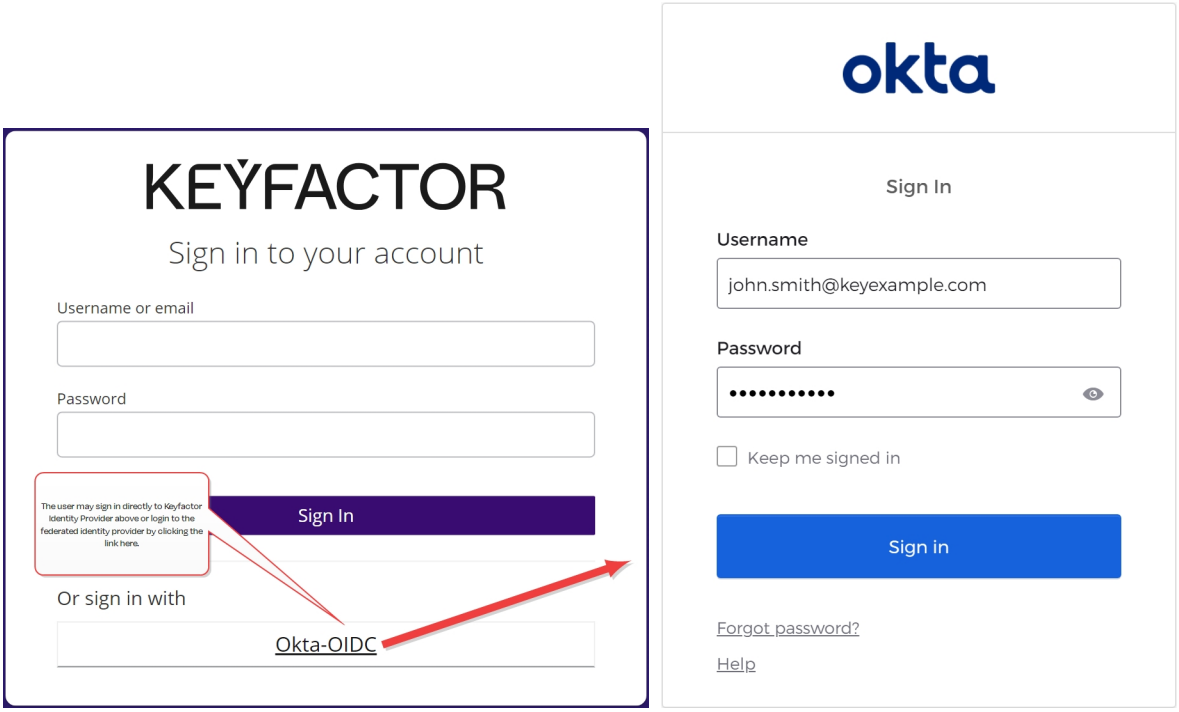


Figure 472: Login Page with Choice of Federated Identity Provider

No changes are needed to the configuration of the OAuth provider in Keyfactor Command when you add federation because all requests are brokered through Keyfactor Identity Provider. At login, the user is presented with a Keyfactor Identity Provider login prompt where he or she can choose to login directly to Keyfactor Identity Provider or click a link that will redirect to the login page of the federated identity provider.

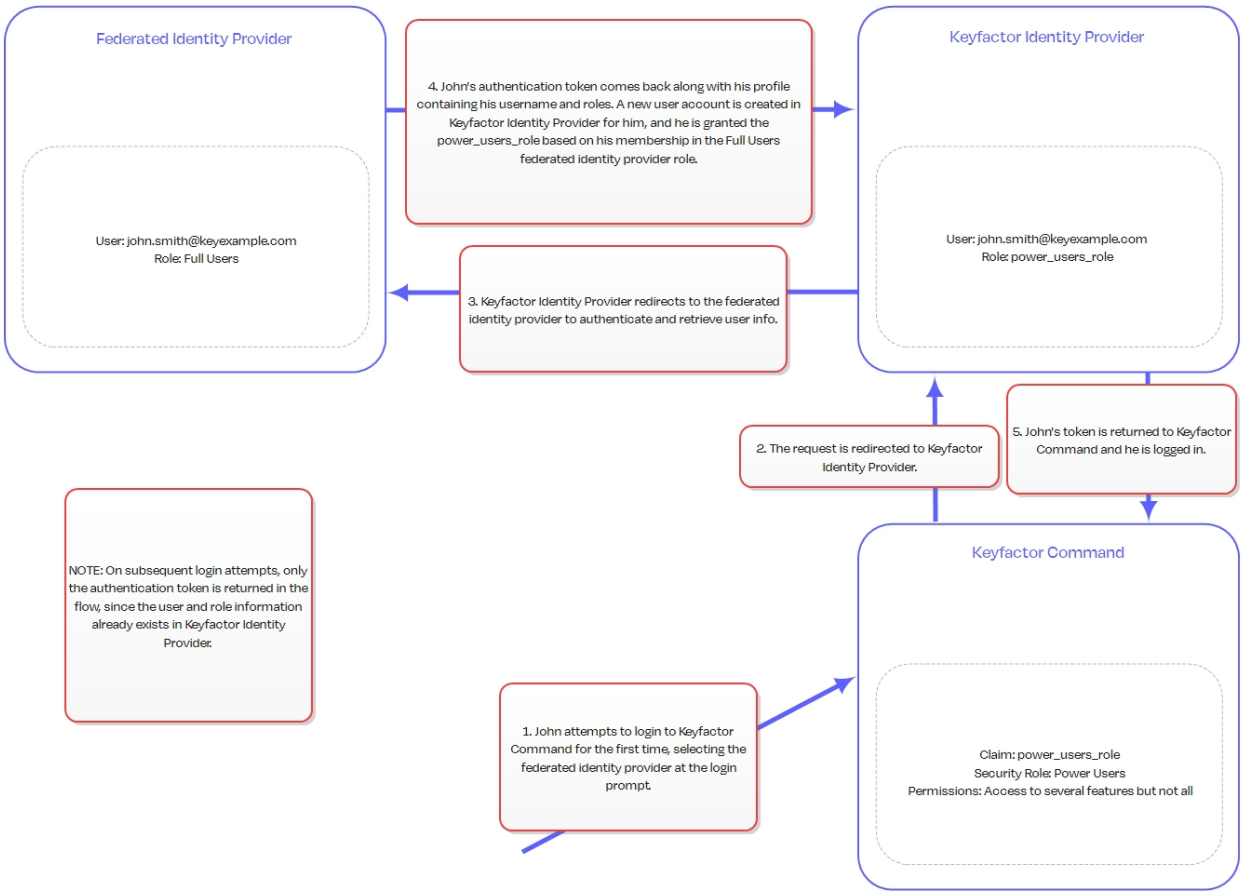


Figure 473: Federated Identity Provider Login Flow

Federating to an Okta OpenID Connect Application

Federating to an Okta OpenID Connect application involves some information gathering and configuration on the Okta side and some configuration in Keyfactor Identity Provider.

To configure the Okta OIDC application and gather information:

1. Login to your Okta management console, browse to *Applications > Applications* and open the details for the application you will be federating. On the General tab, locate your *Client ID* and *Client Secret* and make note of these. You will need these when configuring the federated link from Keyfactor Identity Provider. Confirm that your application is configured to *Require PKCE as additional verification*. Keyfactor strongly recommends the use of this option for best security practice.

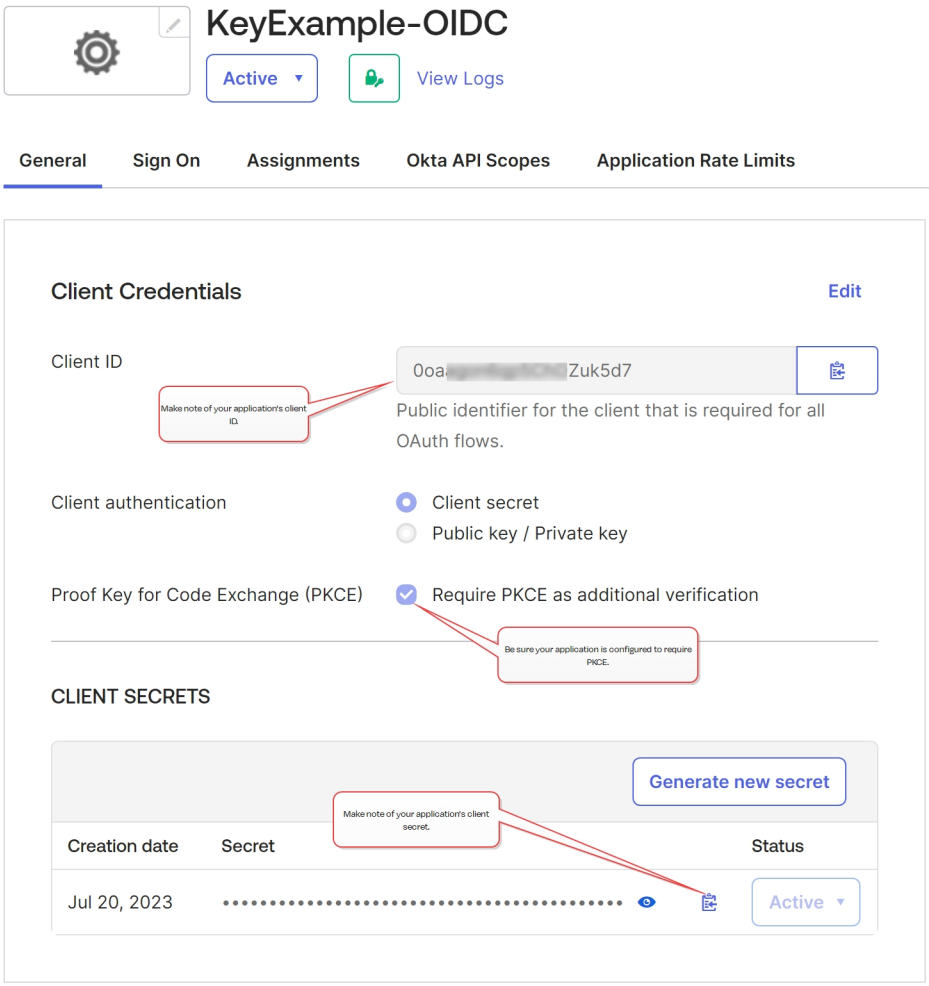


Figure 474: Client ID and Secret in Okta OIDC Application

2. In the Okta OIDC application on the General tab, scroll down to the LOGIN section. Update these values as follows:

    - Sign-in redirect URIs

      The URI for the Keyfactor Identity Provider federated identity provider endpoint. For example:

      ```
      https://appsrvr18.keyexample.com:1443/realms/Keyfactor/broker/Okta-OIDC/endpoint
      ```

      Where appsrvr18.keyexample.com is the fully qualified domain name of the server on which you installed or will install Keyfactor Identity Provider, Keyfactor is the name of the realm in Keyfactor Identity Provider (the default is Keyfactor), and Okta-OIDC is the name you give to the federated identity provider broker you create in Keyfactor Identity Provider to link to your Okta OIDC application.

- Sign-out redirect URIs

  The URI to which users should be redirected on logout. For example:

  ```
  https://appsrvr18.keyexample.com:1443/realms/Keyfactor/broker/Okta-OIDC/endpoint/logout_response
  ```

  Where appsrvr18.keyexample.com is the fully qualified domain name of the server on which you installed or will install Keyfactor Identity Provider, Keyfactor is the name of the realm in Keyfactor Identity Provider (the default is Keyfactor), and Okta-OIDC is the name you give to the federated identity provider broker you create in Keyfactor Identity Provider to link to your Okta OIDC application.

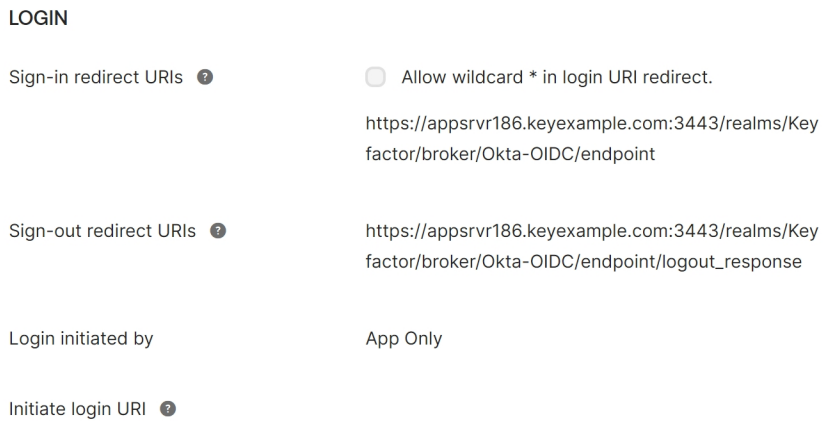


Figure 475: Redirect URIs for the Okta OIDC Application

3. In the Okta management console, browse to *Security > API*, and on the Authorization Servers tab, open the authorization server for your application. In the authorization server details on the Claims tab, click **Add Claim** to create a new claim to send role information to Keyfactor Identity Provider to allow you to map roles in Okta to roles in Keyfactor Identity Provider and then use the roles to assign permissions in Keyfactor Command.

   In the Edit Claim dialog:

   - Give the claim a **Name** to indicate its purpose. You will need to reference this name when creating maps in your Keyfactor Identity Provider federated identity provider.
   - Set **Include in token type** to *ID Token Always*.
   - Select a **Value type** of *Groups*.
   - Enter a **Filter** to control which Okta roles are included with the user information delivered to Keyfactor Identity Provider. To deliver all the groups, you can choose *Matches regex* and enter a regex of .* or use a *Starts with* filter, for example, if all the roles you wish to use with Keyfactor Identity Provider start with the same value (e.g. kyf).
   - Set **Include in** to *Any scope*.

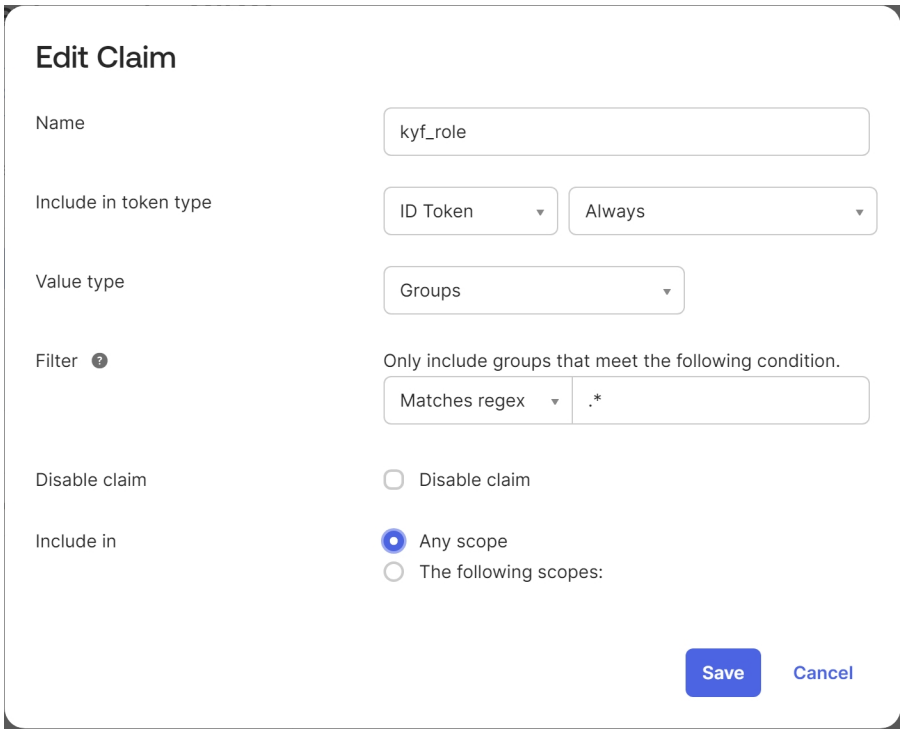


Figure 476: Create an Authorization Server Role Claim

To configure Keyfactor Identity Provider to add a federated identity provider for the Okta OIDC application:

1. Use a browser to open the Keyfactor Identity Provider management interface. For example:

   ```
   https://appsrvr18.keyexample.com:1443
   ```

   Click the **Administration Console** link and sign in with an administrative user and password (see Installing Using Docker Compose on page 2710).

2. In the Keyfactor Identity Provider Administration Console, select Keyfactor in the realm drop-down.

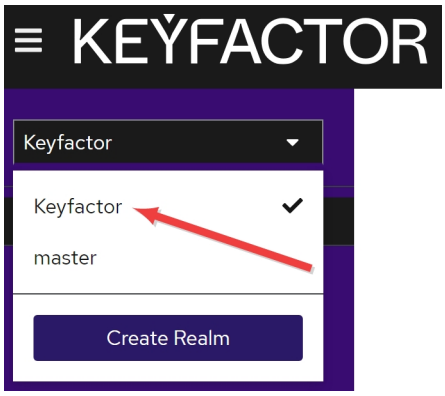


Figure 477: Select a Realm in the Keyfactor Identity Provider Administration Console

3. In the Keyfactor Identity Provider Administration Console, browse to *Identity providers* and click **Add provider**. In the top section, accept the default **Redirect URI**, in the **Alias** field enter the name you used when constructing your redirect URIs in Okta in the previous step, and enter a compatible **Display Name**.

   Tip: The alias field is used as the *identity provider hint* during the configuration of Keyfactor Command if you wish to bypass the Keyfactor Identity Provider login and send users directly to the Okta login.

Figure 478: Give the Keyfactor Identity Provider Identity Provider an Alias

4. In the configuration page for the new provider in the OpenId Connect settings section, choose **Use discovery endpoint** and enter the URL to the discovery endpoint for your application's authorization server. For example:

   ```
   https://${yourOktaDomain}/oauth2/${authorizationServerId}/.well-known/oauth-authorization-server
   ```

Figure 479: Enter the Okta Discovery Endpoint in the Keyfactor Identity Provider Identity Provider

Click **Show metadata** to review the data retrieved from the discovery endpoint.

5. Once the information populates, toggle **Use discovery endpoint** off and click **Show metadata** to display the additional fields. Toggle the **Use PKCE** to *On* and set the **PKCE Method** to *S256* (unless you didn't enable PKCE in Okta).

Figure 480: Enable PKCE in the Keyfactor Identity Provider Identity Provider

6. In the **Client ID** field enter the client ID for your Okta application and enter the secret for your Okta application in the **Client Secret** field.

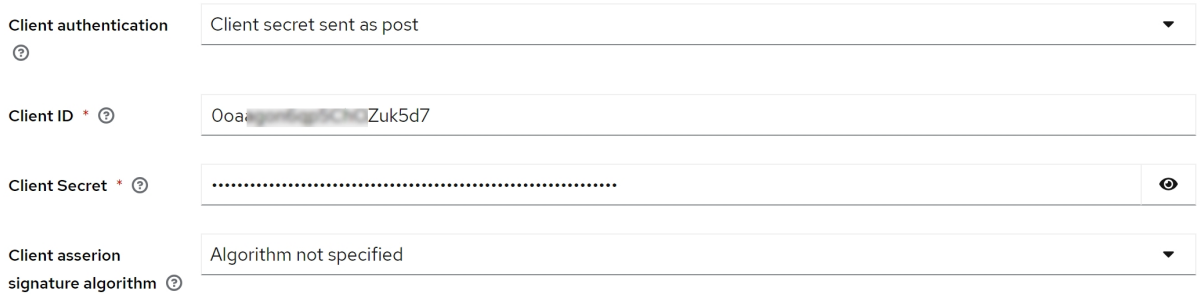


Figure 481: Add Okta Client ID and Secret in the Keyfactor Identity Provider Identity Provider

7. Click **Add** to create the identity provider.
8. Expand Advanced, and in the **Scopes** field add *openid profile*.

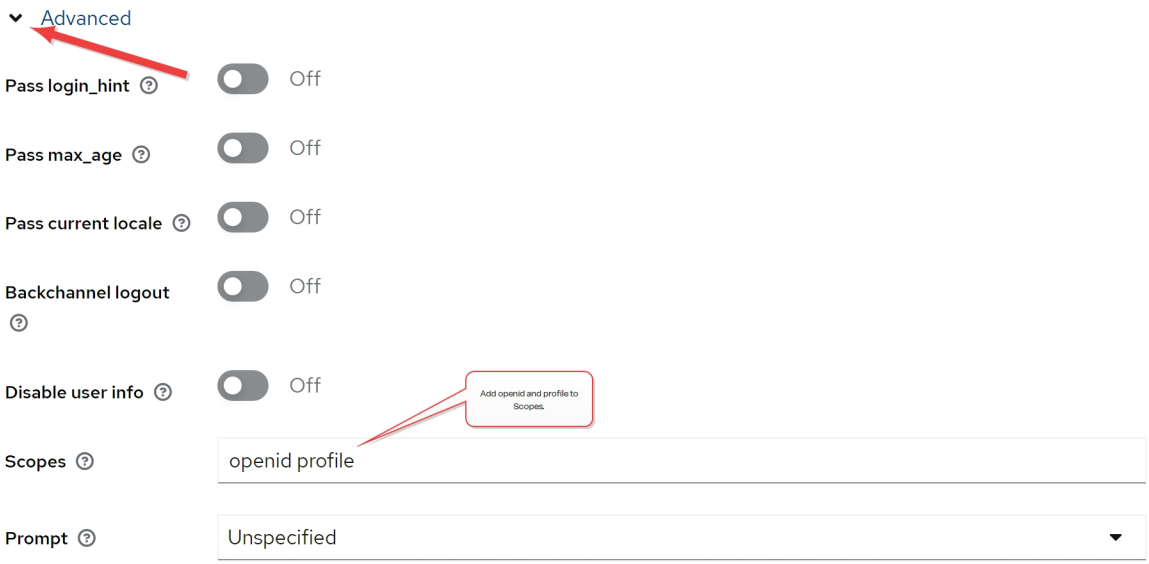


Figure 482: Deliver the Okta openid and profile to the Keyfactor Identity Provider Identity Provider

9. At the top of the identity providers page, switch to the Mappers tab and click **Add mapper**. Here you're mapping the usernames from Okta to the Username field in Keyfactor Identity Provider to allow user records for each Okta user to automatically be generated in Keyfactor Identity Provider.

   On the Add Identity Provider Mapper dialog:

   - Give the mapper a **Name** that will help you identify it.
   - Choose a **Sync mode override** of *Import*.
   - Choose a **Mapper type** of *Username Template Importer*.
   - Set the **Template** to *${CLAIM.preferred_username}*.

     Note: This assumes the value in the Okta *preferred_username* is the one you wish to use as the username for login to Keyfactor Command.

   - Choose a **Target** of *LOCAL*.

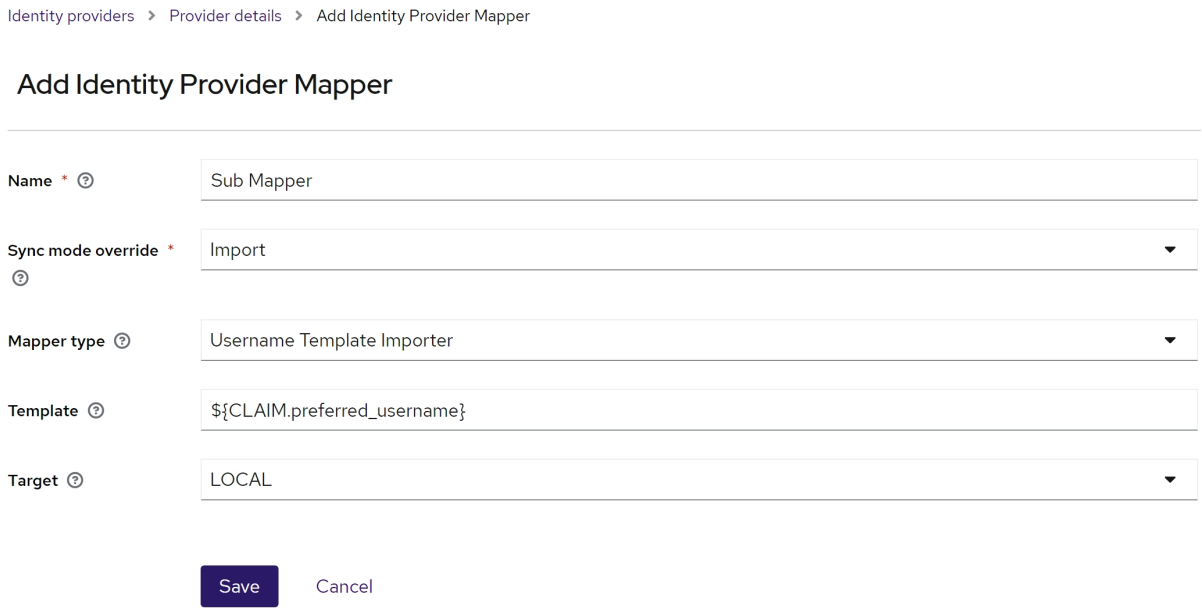


Figure 483: Map the Okta preferred_username to the Keyfactor Identity Provider Identity Provider Username

10. Return to the Provider details and click **Add mapper** again. Here you're mapping the roles from Okta to the roles in Keyfactor Identity Provider to allow the user records for each Okta user in Keyfactor Identity Provider to automatically be assigned roles in Keyfactor Identity Provider.

    On the Add Identity Provider Mapper dialog:

    - Give the mapper a **Name** that will help you identify it. If you'll be adding mappings for more than one Okta group, you may find it helpful to use a consistent naming pattern (e.g. Role Mapper: Admins, Role Mapper: Power Users).
    - Choose a **Sync mode override** of *Import*.
    - Choose a **Mapper type** of *Claim to Role*.
    - Set the **Claim** to the name of the claim you created in Okta to include roles in the claim passed to Keyfactor Identity Provider (e.g. *kyf_role*) as per the Okta steps, above.
    - Click Select Role and choose the role in Keyfactor Identity Provider that should be mapped to the incoming Okta role.

    Note: This step assumes that you've already set up a role in Keyfactor Identity Provider for this purpose (see Using Keyfactor Identity Provider on page 2724).

    Repeat this step for any additional roles from Okta that should be mapped to Keyfactor Identity Provider roles, using the same Claim name for each (all the roles are passed in the same claim). Any roles that come in with the claim that you do not create a mapping for will be ignored.

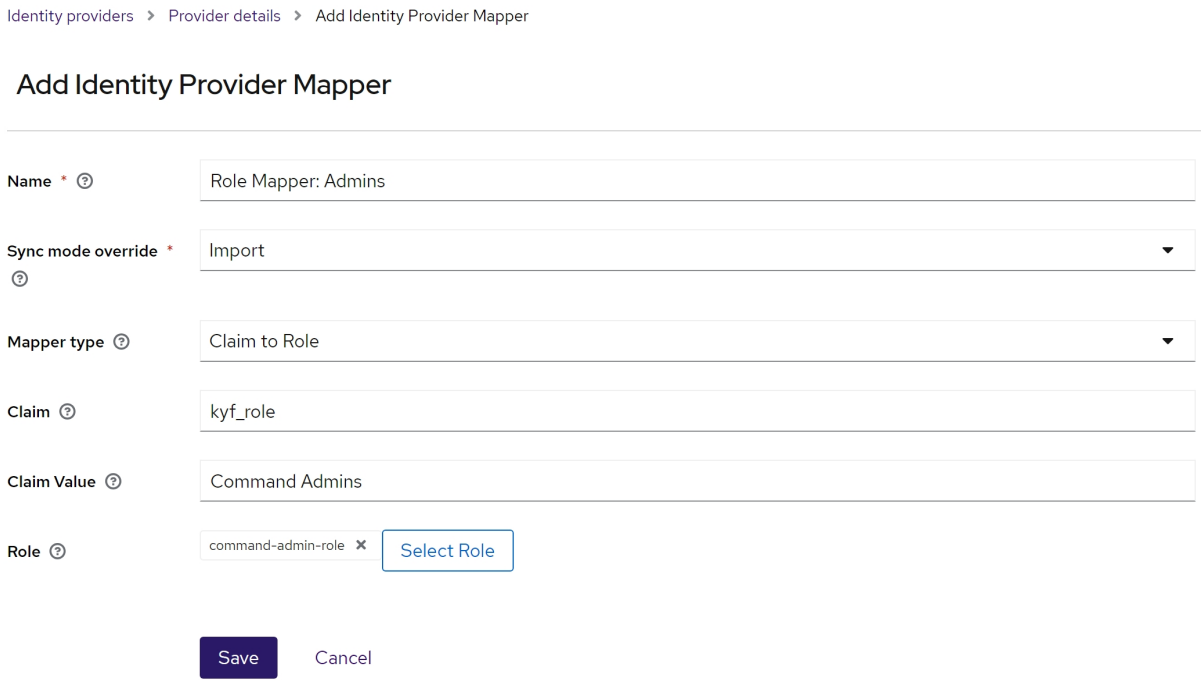


Figure 484: Map the Okta Roles to the Keyfactor Identity Provider Identity Provider Roles

Configuration is complete. No changes are needed to the configuration in Keyfactor Command. It's helpful to restart the web server services (run an iisreset) on the Keyfactor Command server after making the changes to clear any cached data.

4.4.2.2 SQL Server

Keyfactor Command uses a Microsoft SQL Server database to store configuration and synchronized certificate information. Standard edition or above of SQL Server is required. In a production implementation, Keyfactor recommends that SQL Server be installed on a separate server from the Keyfactor Command roles.

Note: Microsoft SQL 2017, 2019, and 2022 all with TLS encryption enabled are supported.

Although you can implement a SQL server especially for Keyfactor Command, in many environments an existing shared SQL server or cluster is used. Keyfactor Command creates one database with a user-defined name and can successfully co-exist with other databases in the same SQL instance.

SQL should be installed with a case-insensitive collation setting.

Connecting to SQL over SSL

By default, Keyfactor Command connects to SQL using an encrypted connection. This requires configuration of an SSL certificate on your SQL server.

If your SQL server is not configured correctly for SSL, you'll see an error message similar to the following when you try to make a connection from Keyfactor Command:

```
Unable to establish a connection to the database server. Please ensure that the server
name is correct and sufficient privileges have been granted to the connection account.:
Encountered an invalid or untrusted certificate and could not connect to the database.
TLS encryption is enabled by default. Please visit 'Planning and Preparing --> SQL
Server' In the Keyfactor Installing Server guide to resolve this.
```

Log message will look something like:

```
2022-09-09 11:35:13.0142 CSS.CMS.Install.ConfigurationWizard.ViewModels.DatabaseViewModel [Error] -
Unable to establish a connection to the database server.  Please ensure that the server name is
correct and sufficient privileges have been granted to the connection account.
2022-09-09 11:35:13.0142 CSS.CMS.Install.ConfigurationWizard.ViewModels.DatabaseViewModel [Error] -
Encountered an invalid or untrusted certificate and could not connect to the database. TLS encryption
is enabled by default. Please visit 'Planning and Preparing --> SQL Server' in the Keyfactor
Installing Server guide to resolve this.
at CSS.CMS.Install.ConfigurationWizard.ViewModels.DatabaseViewModel.a(Object A_0, RunWork-
erCompletedEventArgs A_1)
A connection was successfully established with the server, but then an error occurred during the
login process. (provider: SSL Provider, error: 0 - The certificate chain was issued by an authority
that is not trusted.)
```

To acquire a new SSL certificate or check for an existing certificate, see Using SSL to Connect to SQL Server on page 2746.

If you would prefer not to use an encrypted channel for your connection to SQL, see Configurable SQL Connection Strings on page 2750.

Database Encryption

Keyfactor Command uses Microsoft SQL Server column encryption with the ENCRYPTBYKEY and DECRYPTBYKEY cryptographic functions to protect sensitive data. The type of data protected in this way includes:

- Service account credentials
- SMTP credentials
- Certificate store passwords
- Certificate and pending certificate request private keys
- API secrets
- The 64-byte key used to sign audit log records

SQL encryption is built in to the product and cannot be disabled. In addition to SQL encryption, Keyfactor Command offers optional application-level encryption. This option allows you to encrypt

select sensitive data stored in the Keyfactor Command database using a separate encryption methodology utilizing a Keyfactor Command-defined certificate on top of the SQL server encryption. This additional layer of encryption protects the data in cases where the SQL Server master keys cannot be adequately protected. For more information, see Application-Level Encryption on page 2752.

Database Backup

Backup of the SQL server Database Master Key (DMK) for the Keyfactor Command database is of critical importance in database backup and recovery operations. The backup file of the DMK and the password for it should be stored in a safe, well-documented location. Without the file and password created with this process, some data that is encrypted within the Keyfactor Command database will be unrecoverable in a disaster recovery scenario. For more information, see SQL Encryption Key Backup on page 821.

High Availability

For a highly available solution, Keyfactor recommends using always on availability groups. The availability groups feature of SQL Server sits on top of Windows Server failover clustering and provides the ability to automatically synchronize multiple copies of databases across geographically dispersed SQL Servers. Although the availability groups feature relies on Windows clustering, it does not require shared storage, so it is appropriate for a geo-redundant deployment. The availability groups feature is the current recommended solution from Microsoft. Because Keyfactor Command makes use of SQL database encryption, when availability groups are configured, the Keyfactor Command service master key (SMK) must be synchronized between all participating nodes in the availability group. This can be accomplished by backing up the SMK from one SQL server and restoring it to the other servers in the availability group. For more information, see SQL Encryption Key Backup on page 821.

Grant Permissions in SQL

The user installing Keyfactor Command needs permissions to administer the SQL server and add databases and users (logins) in SQL. Full sysadmin permissions are needed to upgrade from a previous version of Keyfactor Command if the user running the upgrade is not the same user who installed the previous version of Keyfactor Command.

Windows Authentication

If you opt to use Windows authentication for the Keyfactor Command connection to SQL during the installation, the user who installs Keyfactor Command needs appropriate permissions in SQL. To grant this, add the user who will install Keyfactor Command to the SQL server login list:

1. On the SQL server open the SQL Server Management Studio, connect to the database, and open **Security**.
2. Right-click on **Logins** and choose **New Login**.

3. Select the **Windows authentication** radio button.
4. Enter the domain name and user name of the administrative user who will be installing Keyfactor Command in the **Login name** field.
5. On the Login Properties page for this user, open Server Roles and check either the sysadmin role or the dbcreator, public and securityadmin roles, depending on whether this is a new install or an upgrade (see above).
6. Accept the remainder of the defaults and click **OK**.

Once Keyfactor Command has been deployed, the Windows user used for the install can be removed from the Logins under Security in the SQL Server Management Studio. Ongoing connectivity to the database is maintained using logins automatically created in SQL for the Keyfactor Command application pool users (see Create Service Accounts for Keyfactor Command on page 2757) specifically for the purpose during the installation.

SQL Authentication

If you opt to use SQL authentication, appropriate permissions need to be granted to the SQL user entered in the initial connection dialog of the Keyfactor Command Configuration Wizard. You may choose to create (or use an existing) SQL user for the installation and create a separate SQL user for ongoing connectivity or use the same user for both purposes.

Note: Your SQL server must be configured to support mixed mode authentication in order to use the SQL authentication option.

To create a new SQL user for the initial SQL connection:

1. On the SQL server open the SQL Server Management Studio, connect to the database, and open **Security**.
2. Right-click on **Logins** and choose **New Login**.
3. Select the **SQL Server authentication** radio button.
4. Enter a user name for the SQL user in the **Login name** field and enter and confirm a **Password**. You may wish to uncheck the **User must change password at next login** box.
5. On the Login Properties page for this user, open Server Roles and check either the sysadmin role or the dbcreator, public and securityadmin roles, depending on whether this is a new install or an upgrade (see above).
6. Accept the remainder of the defaults and click **OK**.

Once Keyfactor Command has been deployed, this SQL user may be removed if it is not also serving the role of providing ongoing connectivity. During installation, you enter a SQL user name and password for a login to maintain ongoing connectivity. If this login already exists in SQL, it will be granted appropriate permissions. If this login does not already exist in SQL, it will be created and granted appropriate permissions.

Note: Automatically generated service accounts are not created with the db_owner role. Instead, a keyfactor_db_role is created and granted to the service accounts. This role has permission on each of the schemas (dbo, ssl, ssh, cms_agents, etc.) and permission on the encryption certificate.

Using SSL to Connect to SQL Server

By default, Keyfactor Command connects to SQL using an encrypted connection using an SSL certificate configured on your SQL server.

You can check whether your SQL server has been configured with an SSL certificate in one of two ways:

SQL Server Configuration Manager

1. On the SQL server, open the SQL Server Configuration Manager and drill down under SQL Server Network Configuration to find *Protocols for [YOUR INSTANCE NAME]*.
2. Right-click on Protocols for [YOUR INSTANCE NAME] and choose **Properties**.
3. Check the Certificate tab of the Properties dialog to see if a certificate has been configured and is still valid. If your certificate has a friendly name, it will appear here listed in the dropdown by its friendly name.

Important: A certificate will only appear here if it has a CN[1], usually the FQDN of the SQL server. If a certificate has been configured without this, it will not appear to be configured through this UI.

[1]The Subject property of the certificate must indicate that the common name (CN) is the same as the host name or fully qualified domain name (FQDN) of the server computer or it must match the DNS suffix if using a wildcard certificate. When using the host name, the DNS suffix must be specified in the certificate. If SQL Server is running on a failover cluster, the common name must match the host name or FQDN of the virtual server and the certificates must be provisioned on all nodes in the failover cluster.

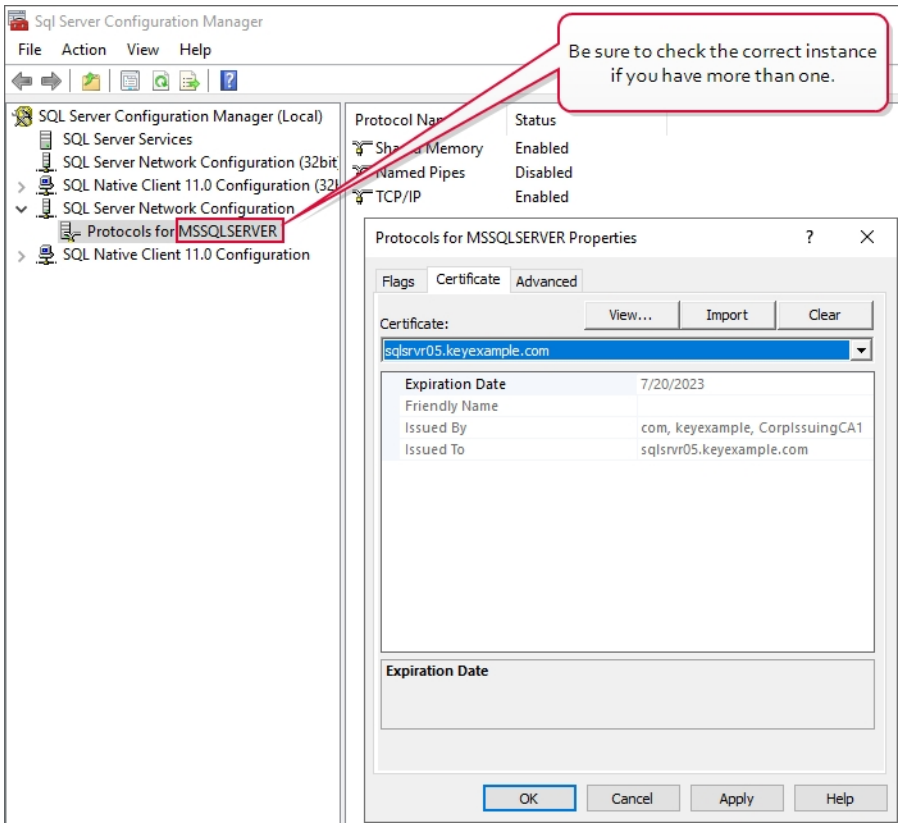


Figure 485: SQL Server Configuration Manager View Active SSL Certificate

Registry

1. On the SQL server, open the registry editor and browse to (where *[MSSQL15.MSSQLSERVER]* is the correct version of SQL server for your server):

    ```
    HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\
    [MSSQL15.MSSQLSERVER]\MSSQLServer\SuperSocketNetLib
    ```

2. In the SuperSocketNetLib registry key, look for a Certificate value.

3. Validate that the Certificate value has a thumbprint configured. This should match the thumbprint of an active certificate with a Server Authentication EKU in the Local Machine certificate store.

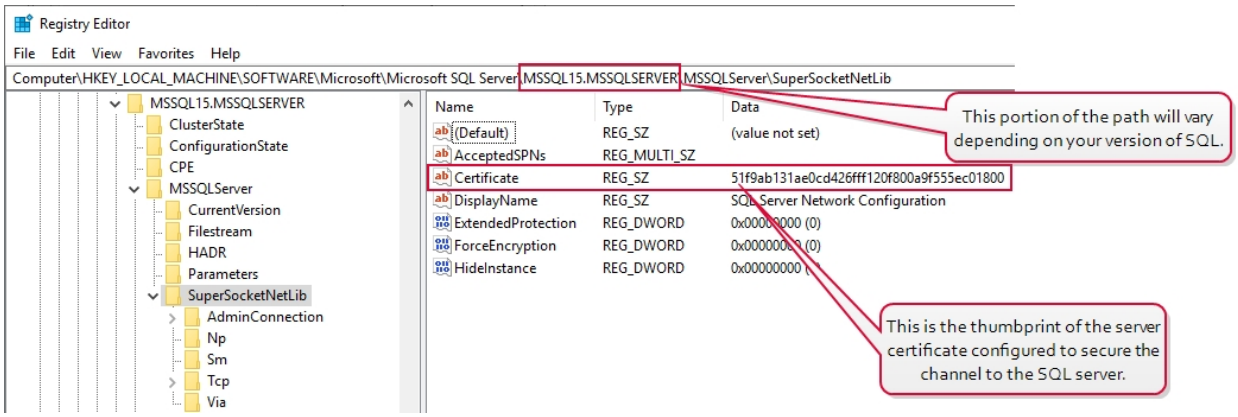


Figure 486: Registry View Active SSL Certificate

To acquire a new certificate for your SQL server:

1. On the SQL server open the Services.msc MMC and scroll down to locate the *SQL Server ([YOUR INSTANCE NAME])* service.
2. Check the *Log On As* column for the name of the service account that the service is running as.

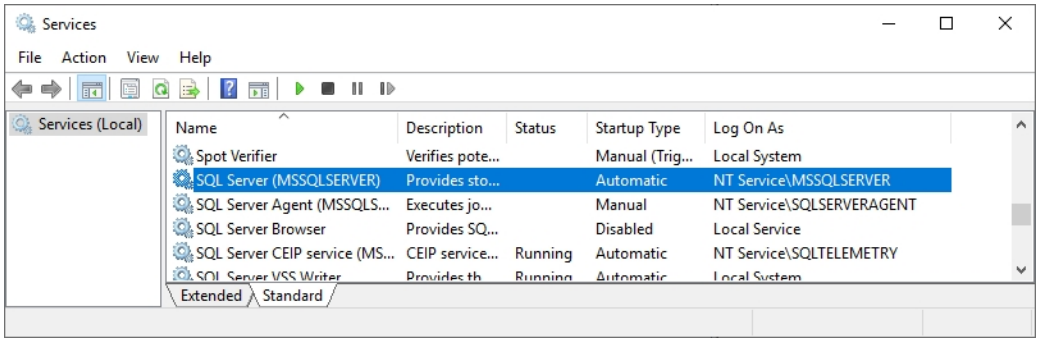


Figure 487: View SQL Server Services

3. Identify a template with a Server Authentication EKU (a typical web server template).
4. On the SQL server, do one of following:
    - Using the GUI:
        a. Open an empty instance of the Microsoft Management Console (MMC).
        b. Choose **File->Add/Remove Snap-in...**.
        c. In the *Available snap-ins* column, highlight **Certificates** and click **Add**.
        d. In the Certificates snap-in popup, choose the radio button for Computer account, click **Next**, accept the default of Local computer, and click **Finish**.
        e. Click **OK** to close the Add or Remove Snap-ins dialog.

- Using the command line:
    a. Open a command prompt using the “Run as administrator” option.
    b. Within the command prompt type the following to open the certificates MMC:

        certlm.msc

5. Enroll for the certificate using your preferred method, being sure to give the certificate a CN (it will not appear in the configuration tool without this) and add subject alternative names (SANs) to it for all the IP addresses, server names, and FQDNs that you might use to reference the SQL server when communicating with it, including DNS aliases. Install it, along with its private key, into the Local Machine certificate store on the SQL server. One way to do this is in the certificates MMC:
    a. Drill down to the Personal folder under **Certificates** for the Local Computer, right-click, and choose **All Tasks->Request New Certificate...**.
    b. Follow the certificate enrollment wizard, selecting the template you identified for this purpose, and providing appropriate SANs along with any required information.

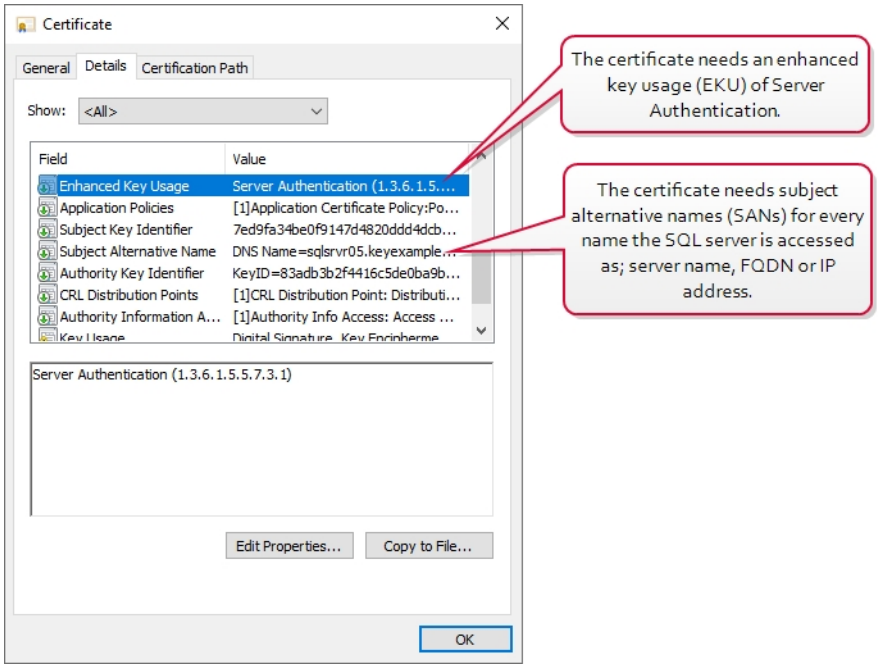


Figure 488: SQL Server SSL Certificate Details

6. Drill down to the Personal folder under **Certificates** for the Local Computer, locate your certificate, right-click, and choose **All Tasks->Manage Private Keys...**.
7. In the Permissions for private keys dialog, click **Add**, add the SQL service account, and grant that service account **Read** but not **Full control** permissions. If the SQL server is running as

NT Service\[YOUR INSTANCE NAME] as shown in Figure 487: View SQL Server Services, be sure to change the location to your local machine and enter the object name as "NT SERVICE\[YOUR INSTANCE NAME]" as shown in Figure 489: Grant Private Key Permissions for SQL Server.

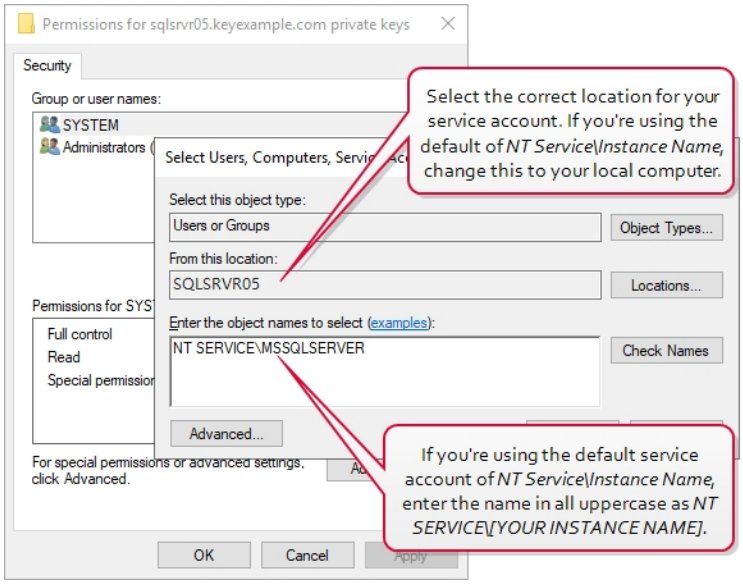


Figure 489: Grant Private Key Permissions for SQL Server

8. Click **OK** to save.
9. Configure the SSL certificate in SQL using either the SQL Server Configuration Manager or registry as shown above for checking whether there is an existing certificate configured (see SQL Server Configuration Manager on page 2746).
10. After you've acquired a new certificate, made the private key permission changes, and associated it in SQL, you'll need to restart the *SQL Server (Instance Name)* service (see Figure 487: View SQL Server Services) before these changes will take effect.

Configurable SQL Connection Strings

Keyfactor Command supports using a template SQL connection string that can be created to fit the needs of the overall deployment. This template will be used as a starting point and will not be overwritten by the configuration wizard. For instance, you can set the timeout setting in one place, and once the configuration wizard is run, this is reflected in all places where a connection string is used. The template can be changed at any time to update the connection strings.

To create a customized connection string, after installing the Keyfactor Command software but before running the configuration wizard, modify both the EFModels and SqlDirect connection strings in the **SqlConnectionStrings.json** file found in the *Configuration* folder under your installation directory. By default, this is:

C:\Program Files\Keyfactor\Keyfactor Platform\Configuration

The settings that can be modified are described in the following Microsoft article:

https://docs.microsoft.com/en-us/dot-net/api/system.data.sqlclient.sqlconnection.connectionstring?view=dotnet-plat-ext-6.0

Note: The *Data Source*, *Initial Catalog*, *Integrated Security*, *User ID* and *Password* settings are reserved for the configuration wizard to use to configure and save the authentication for the connection string, but other settings found in the template string are left as-is.

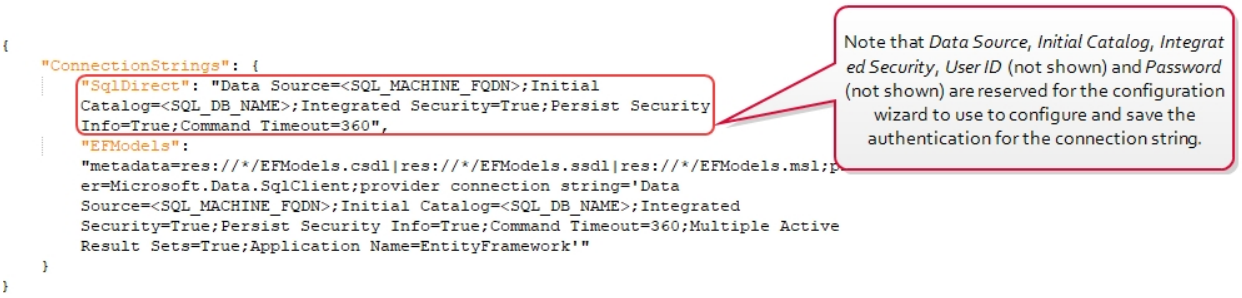


Figure 490: Default SQL Connection Strings

Disable Encryption

If you prefer to connect to your SQL server over a non-encrypted channel (and thus avoid configuring an SSL certificate for your SQL server), you can use the *Encrypt* keyword in the connection strings with a value of *False*.

```
{
    "ConnectionStrings": {
        "SqlDirect": "Data Source=<SQL_MACHINE_FQDN>;Initial
        Catalog=<SQL_DB_NAME>;Integrated Security=True;Persist Security
        Info=True;Command Timeout=360",
        "EFModels":
        "metadata=res://*/EFModels.csdl|res://*/EFModels.ssdl|res://*/EFModels.msl;provid
        er=Microsoft.Data.SqlClient;provider connection string='Data
        Source=<SQL_MACHINE_FQDN>;Initial Catalog=<SQL_DB_NAME>;Integrated
        Security=True;Persist Security Info=True;Command
        Timeout=360;Encrypt=False;Multiple Active Result Sets=True;Application
        Name=EntityFramework'"
    }
}
```

Figure 491: SQL Connection Strings with Encrypt Channel Disabled

Use a SQL Server Listening on Multiple IP Addresses

If you're using a SQL server cluster that's configured to listen for incoming connections on more than one IP address to support redundancy or access from multiple networks/subnets, you can use

the *MultiSubnetFailover* keyword in the connection strings with a value of *True*.

```
{
    "ConnectionStrings": {
        "SqlDirect": "Data Source=<SQL_MACHINE_FQDN>;Initial
        Catalog=<SQL_DB_NAME>;Integrated Security=True;Persist Security Info=True;Command
        Timeout=360",
        "EFModels":
        "metadata=res://*/EFModels.csdl|res://*/EFModels.ssdl|res://*/EFModels.msl;provider
        =Microsoft.Data.SqlClient;provider connection string='Data
        Source=<SQL_MACHINE_FQDN>;Initial Catalog=<SQL_DB_NAME>;Integrated
        Security=True;Persist Security Info=True;Command
        Timeout=360;MultiSubnetFailover=True;Multiple Active Result Sets=True;Application
        Name=EntityFramework'"
    }
}
```

Figure 492: SQL Connection Strings with MultiSubnetFailover Option Enabled

Application-Level Encryption

Keyfactor Command uses data encryption for sensitive data—such as private keys for certificates—stored in the Keyfactor Command database (see SQL Server on page 2742). This option encrypts only the data in the database deemed to be of a sensitive nature, not the entire database. By default, the data is encrypted using SQL encryption, but you have the option to add another level of security with application-level encryption. If you choose to enable this option, you will need a certificate for this purpose installed in the Personal Certificate store of the Local Computer on each Keyfactor Command server. The certificate must have a key usage of either key encipherment or data encipherment enabled. Microsoft certificate templates only allow you to configure data encipherment ("Allow encryption of user data") as a suboption to key encipherment ("Allow key exchange only with key encryption"). You do not need to enable both. You may use the certificate acquired in the name of the Keyfactor Command web site (see Acquire a Public Key Certificate for the Keyfactor Command Server on page 2767), assuming it supports the appropriate key usage, or you may enroll for a separate certificate for this purpose. The same certificate must be used on all Keyfactor Command servers and the certificate must be available in the certificate store on the machine when you run the Keyfactor Command installation. A hardware security module (HSM) may be used, if desired. To support the use of an HSM, the Windows CSP driver for the HSM must be installed on the Keyfactor Command server. Be aware that transactions accessing the encrypted data—such as enrolling for PFX certificates, downloading PFX certificates, running inventory, and adding certificates to certain types of certificate stores (e.g. F5, NetScaler)—will require accessing the HSM. A slow HSM will slow down these processes.

Note: In an environment where there are multiple copies of Keyfactor Command pointing to the same database, each server running a Keyfactor Command instance will need to have the same encryption certificate AND the corresponding private key.

Note: The thumbprint of the certificate used for application-level encryption is stored in the registry on the Keyfactor Command server(s)—rather than in the Keyfactor Command database—to provide a further level of separation from SQL.

Important: If the certificate used for application-level encryption or the private key for this certificate are removed from the Keyfactor Command server while data in the database is encrypted with this certificate, access to this data will be lost. Take care to ensure that this certificate and its private key remain in place or that there are backups of both the certificate and private key (with any necessary password) that can be accessed in the event that the certificate needs to be restored.

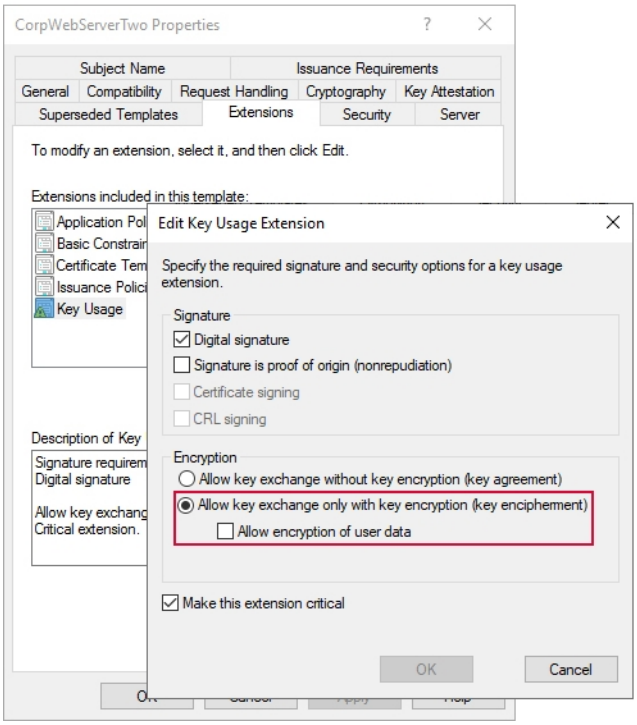


Figure 493: Certificate Template with Key Encipherment Key Usage

Cryptographic Provider

The certificate you use for application-level encryption must be issued using a key storage provider (KSP) or at least installed on the Keyfactor Command server using a KSP in order to allow Keyfactor Command to grant permissions on the certificate's private key appropriately during installation. If Keyfactor Command is not able to access the private key of the application-level certificate, you may receive a 500.30 error trying to access the Management Portal or you may find you can access the Management Portal but not all features function and you find errors such as these in the log (which of these occurs depends on other configuration factors):

```
Unable to decrypt enveloped PKCS7 data
```

```
Keyset does not exist
```

You can check the provider on your certificate with a command similar to the following issued in an administrative command prompt:

```
certutil -store MY [thumbprint of the application encryption certificate]
```

Output from this command should like similar to the following for a KSP certificate:

```
MY "Personal"
================ Certificate 3 ================
Serial Number: 1800000f39f4c506c41239c566000200000f39
Issuer: CN=CorpIssuingCA1, DC=Keyexample, DC=com
NotBefore: 10/27/2023 4:19 PM
NotAfter: 10/26/2025 4:19 PM
Subject: CN=keyfactor.keyexample.com
Non-root Certificate
Template: CorpWebServerv2, Corp Web Server v2
Cert Hash(sha1): 89b5099bc1f7146185331017db60373afb136edb
   Key Container = te-CorpWebServerv2-8b4d6ca7-e5ec-472f-a096-8b4aa590b22b
   Unique container name: 153703d13e6c7339e297f44547260e6d_00139f2c-9f21-4793-b740-bb3fe658245c
   Provider = Microsoft Software Key Storage Provider
Private key is NOT exportable
Encryption test passed
CertUtil: -store command completed successfully.
```

Output from this command should like similar to the following for a CSP certificate (legacy CSPs vary):

```
MY "Personal"
================ Certificate 0 ================
Serial Number: 59000006513beacf07ce121e45000100000651
Issuer: CN=CorpIssuingCA1, DC=Keyexample, DC=com
NotBefore: 10/27/2023 4:32 PM
NotAfter: 10/26/2025 4:32 PM
Subject: CN=keyfactor.keyexample.com
Non-root Certificate
Template: CorpWebServer, Corp Web Server
Cert Hash(sha1): a3a1299d3f5d209c89573c356495547b67d92f15
   Key Container = d5549bc8ea7af0f51d8b26ffbe9617b8_00139f2c-9f21-4793-b740-bb3fe658245c
   Simple container name: te-CorpWebServer-c6c249ac-66d3-427e-aff0-8de81250887f
   Provider = Microsoft RSA SChannel Cryptographic Provider
Private key is NOT exportable
Encryption test passed
CertUtil: -store command completed successfully.
```

If you have a certificate in PFX format with CSP and would like to import it as a KSP, you can use a command similar to the following:

```
certutil -importPFX -csp KSP MY <PFX Path and FileName> NoExport
```

Alternately, if you're unable to use a certificate with a KSP, you may use a CSP and manually grant the Keyfactor Command application pool user(s) private key read permissions on the certificate (see Using SSL to Connect to SQL Server on page 2746).

4.4.2.3 Certificate Authorities

In most cases, if you are installing Keyfactor Command then you have at least one Microsoft or EJBCA Certificate Authority (CA) in your environment. As you're planning for Keyfactor Command, you'll need to make the following decisions about the CA(s) and certificate templates for your environment:

- Which CAs should be synchronized to the Keyfactor Command database?

  Your license may not allow you to synchronize all of your CAs. Certificates belonging to offline root or policy CA *chain* certificates can be monitored without impacting your license.

  > **Note:** Keyfactor Command contains a constraint that prevents any two certificate authorities from having the same logical name and host name combination. Think about the logical name and host name configuration of the CAs that will be implemented with Keyfactor Command and check for duplicates.

- If you have Microsoft CAs, what authorization method will you use to configure the CAs (see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832)?
- If your Keyfactor Command license includes certificate enrollment:
  - Which CA(s) will be used to issue certificates based on CSRs through the Keyfactor Command Management Portal?
  - Which CA(s) will be used to issue PFXs through the Keyfactor Command Management Portal?
  - Which certificate template(s) will be used to generate CSRs through the Keyfactor Command Management Portal? In most cases, these templates will already exist.
  - Which certificate template(s) will be configured for CSR enrollment in the Keyfactor Command Management Portal? In most cases, these templates will already exist.
  - Which certificate template(s) will be configured for PFX enrollment in the Keyfactor Command Management Portal? In most cases, these templates will already exist.
- If your Keyfactor Command license includes Mac auto-enrollment, which CA(s) will be used to automatically issue certificates to Macs running the Mac auto-enrollment agent?

  As part of the Keyfactor Command installation preparation, you may need to create a certificate template for this purpose.

Tip: This information is not needed to complete the initial installation and configuration, but will be needed to do post-installation configuration in the Keyfactor Command Management Portal.

4.4.2.4 Keyfactor Command Server(s)

A Keyfactor Command server implementation is made up of several Keyfactor Command roles:

Keyfactor Command Management Portal

The server with this role provides the web-based administration interface that is used to view and report on certificates issued in the environment and enroll for certificates. This role runs under Microsoft IIS. Configuration for the Keyfactor Command implementation as a whole is also done through the Keyfactor Command Management Portal. The Logi Analytics Platform for reporting is hosted on the server with this role.

This role is required on all Keyfactor Command servers.

Keyfactor Command Windows Services

The server with this role hosts back-end services required to support Keyfactor Command. This includes the Keyfactor Command Service, which is used for all periodic tasks throughout Keyfactor Command, including CA synchronization, monitoring alerts, and report automation.

This role is required on all Keyfactor Command servers.

Keyfactor Command Web API

The server with this role hosts the Keyfactor API. The Keyfactor API is also included in the Management Portal role, since the Management Portal makes extensive use of this API.

This role is optional. If you choose not to install this role, you will still be able to use the Keyfactor API. This role is available as a separate component for users who wish to install the Keyfactor API on a separate server from the Management Portal server.

Keyfactor Command Orchestrator Service API

The server with this role hosts the back-end service for receiving requests from and sending requests to Keyfactor agents and orchestrators.

This role is optional. If you choose not to install this role, you will not be able to use agents and orchestrators with Keyfactor Command.

In many environments, the Keyfactor Command Management Portal, Windows Services, Web API, and Orchestrator Service API roles are collocated on a single server (or pair of servers if redundancy is desired). Both physical and virtual servers are supported.

Tip: See Install: Select Components on page 2782 for related information.

For a high availability (HA) solution using the same roles on all nodes, note that the following conditions apply:

- All servers must point to the same Keyfactor Command SQL database.
- All servers must be configured with the same encryption certificate AND the corresponding private key (see Database Tab on page 2799).
- Keyfactor recommends that the Keyfactor Command Service be configured to run all services on each node. This allows the service to manage the jobs most efficiently—the service will check out jobs via a locking mechanism that will enforce that any jobs are running on only one service at a time. However, you do have the option to manually tune the jobs on the servers if desired (such that server A always does jobs 1, 2 and 3 and server B always does jobs 4, 5 and 6).
- Review load balancing rules and configuration, if applicable. Load balancing configuration is beyond the scope of this guide.

Keyfactor does not recommend installing any of these roles on a CA or on a SQL server in a production environment.

As you plan for Keyfactor Command, you need to decide upon an architecture for the implementation and prepare servers with sufficient resources accordingly. See System Requirements on page 2702 for more information about planning for servers with sufficient resources to support the planned roles.

Licensing

The Keyfactor Command product is licensed by component. Your license for Keyfactor Command may not include all the features described in this guide. If you choose to add additional components to your Keyfactor Command license in the future, these features can generally be configured without the need to reinstall Keyfactor Command.

4.4.2.5 Create Service Accounts for Keyfactor Command

Several of the Keyfactor Command roles operate under a service account. You can either create a single service account for all these roles or create separate service accounts for each role. If multiple Keyfactor Command roles will be installed on the same server, some of the below roles will be redundant.

> ⚠ **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

The roles that require a service account are:

Keyfactor Command Installer

The user who runs the Keyfactor Command installation must have local administrator permissions on the Keyfactor Command server(s) and must be granted permissions in SQL if Windows authentication for SQL will be used during the installation (see Grant Permissions in SQL on page 2744). You can either grant these permissions to an existing user or you can create a Keyfactor Command installer account and grant the appropriate permissions to this account.

Additionally, the user installing Keyfactor Command must have the SeBackupPrivilege and SeRestorePrivilege rights on the Keyfactor Command server. Normally, administrators are granted these permissions by default, but you should confirm the permissions prior to starting the install. These permissions can be set through Group Policy or Local Security Policy, and can be found under "Local Policies\User Rights Assignment" as "Back up files and directories" and "Restore files and directories".

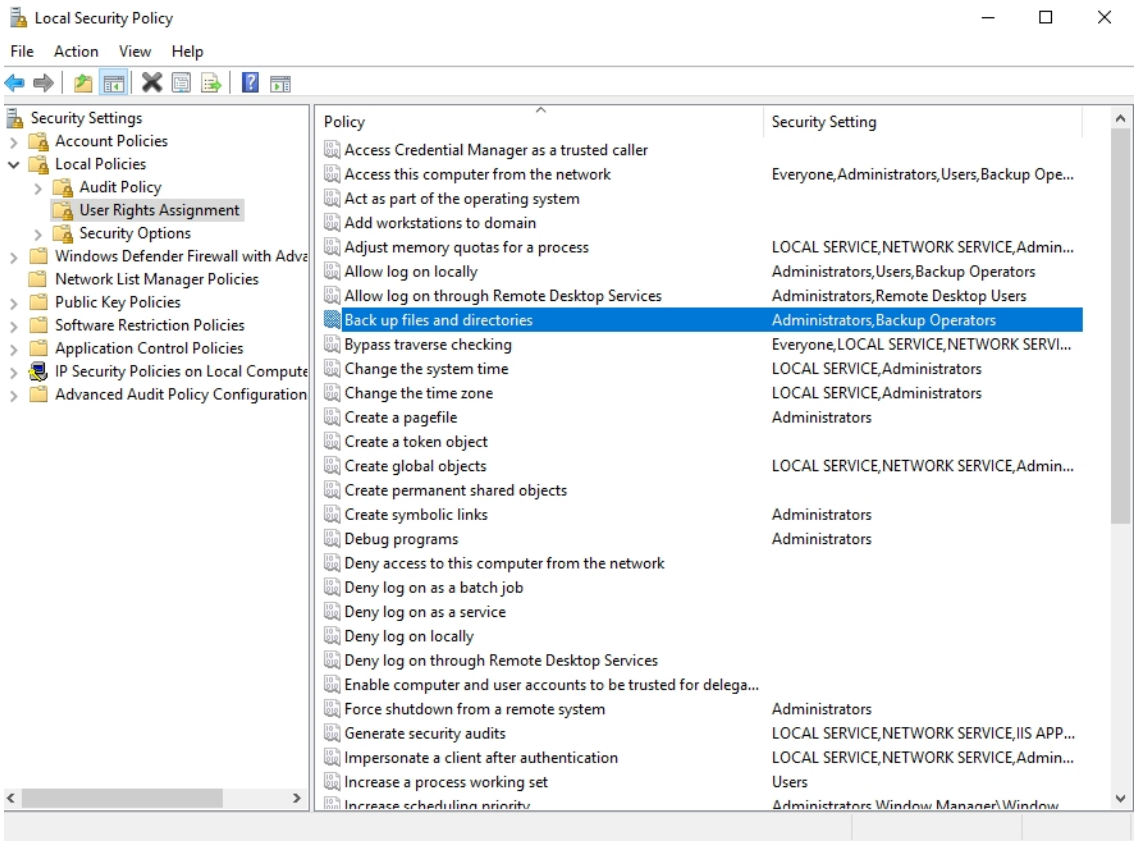


Figure 494: Local Security Policy

For more information on this from Microsoft, see:

> https://docs.microsoft.com/en-us/windows/win32/api/userenv/nf-userenv-loaduserprofilea#remarks

Keyfactor Command Service

The Keyfactor Command Service (a.k.a. the timer service) runs on the Keyfactor Command services server. It synchronizes certificates to the SQL database and initiates notification and reporting tasks. This service runs in the context of an Active Directory or local service account.

The user with this role will be granted permission on each of the SQL schemas (dbo, ssl, ssh, cms_agents, etc.) and permission on the encryption certificate in SQL through the keyfactor_db_role which is created during configuration.

The user with this role must have the "Log on as a service" right on the Keyfactor Command server. Normally, this permission is granted automatically as part of the installation process. You can confirm the permissions through Group Policy or Local Security Policy in "Local Policies\User Rights Assignment". Validate that the user associated with the Keyfactor Command Service has been added to "Log on as a service" directly or indirectly (via group membership).

The user with this role needs to be able to create log files and write to them. During installation, this permission is granted by granting "Create files / write data" and "Create folders / append data" permissions on the log directory (C:\Keyfactor\logs) to the local users group on the assumption that the local users group will contain either "NT AUTHORITY\authenticated users" or "DOMAIN\Domain Users" and that the service account user will be granted permissions via at least one of these. If this is not the case, permissions for the service account user will need to be granted manually to the log directory.

The user with this role needs to be granted permissions on any certificate authorities from which certificates will be synchronized. Additional certificate authority permission may be needed depending on the features that will be used. For more information, see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832.

Keyfactor Command Management Portal (Application Pool)

The Keyfactor Command Management Portal uses an application pool under IIS to operate. This application pool runs in the context of an Active Directory or local service account.

The user with this role will be granted permission on each of the SQL schemas (dbo, ssl, ssh, cms_agents, etc.) and permission on the encryption certificate in SQL through the keyfactor_db_role which is created during configuration.

The user with this role must have the "Log on as a batch job" and "Impersonate a client after authentication" rights on the Keyfactor Command server. In a typical IIS installation, these rights are granted to the IIS_IUSRS group and the user running any application pool created in IIS inherits these rights without being added to the IIS_IUSRS group. For more information about the IIS_IUSRS group, see:

> https://learn.microsoft.com/en-us/iis/get-started/planning-for-security/understanding-built-in-user-and-group-accounts-in-iis

You can confirm the permissions or set them manually for the application pool user through Group Policy or Local Security Policy in "Local Policies\User Rights Assignment". Validate that either the IIS_IUSRS group or the user associated with the Keyfactor Command application pool has been

added to “Log on as a batch job” and “Impersonate a client after authentication” directly or indirectly (via group membership).

The user with this role needs to be able to create log files and write to them. During installation, this permission is granted by granting “Create files / write data” and “Create folders / append data” permissions on the log directory (C:\Keyfactor\logs) to the local users group on the assumption that the local users group will contain either “NT AUTHORITY\authenticated users” or “DOMAIN\Domain Users” and that the service account user will be granted permissions via at least one of these. If this is not the case, permissions for the service account user will need to be granted manually to the log directory.

The user with this role needs to be granted permissions on any certificate authorities from which certificates will be synchronized. Additional certificate authority permission may be needed depending on the features that will be used. For more information, see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832.

Note: If you’re using Active Directory as an identity provider, the Application Pool account must have read permission on any groups being created. This will allow Keyfactor Command to query for group membership on the groups.

Logi Report Access

The Logi Analytics Platform uses a service account to allow Logi to connect to Keyfactor Command via the Keyfactor API to display the dashboard information. This uses an application pool under IIS to operate. The application pool runs in the context of an Active Directory or local service account. A separate application pool is needed for this service, though the same service account may be used for both.

Keyfactor Command Orchestrators API

The Keyfactor Command Orchestrators API IIS application accepts connections from Keyfactor Command orchestrators and uses an application pool under IIS to operate. This application pool runs in the context of an Active Directory or local service account. If this role will be installed on the server hosting the Keyfactor Command Management Portal role, a separate application pool is needed for this service, though the same service account may be used for both.

Keyfactor Command Keyfactor API

The Keyfactor Command Keyfactor API uses an application pool under IIS to operate. This application pool runs in the context of an Active Directory or local service account. The Keyfactor API is an integral part of Keyfactor Command and is not an optional installation. The Keyfactor API can be configured to support custom applications. If the Keyfactor Command Keyfactor API role will be installed on the server hosting the Keyfactor Command Management Portal role, a separate application pool is needed for this service, though the same service account may be used for both.

Keyfactor Command Claims Proxy

For implementations using an identity provider other than Active Directory, the claims proxy proxies authentication tokens between the Management Portal and the Keyfactor API to enable communications between the portal and the API. This application pool runs in the context of an Active Directory or local service account. A separate application pool is needed for this service, though the same service account may be used for both.

This role does not exist in implementations using Active Directory as an identity provider.

Substitutable Text Token Query Access

Keyfactor Command supports the use of substitutable special text tokens in workflow and alert emails, conditions, and data manipulation steps to replace a variable with data from the certificate request, certificate, or certificate metadata at processing time. For tokens that relate to the certificate requester, this involves querying the identity store to retrieve information about the requester. If you're using Active Directory as an identity provider, this query is done in the context of the Keyfactor Command Management Portal application pool user (see Keyfactor Command Management Portal (Application Pool) on page 2759), which must be running as an Active Directory user in that case. If you're using an identity provider other than Active Directory, queries cannot be completed to the identity provider for substitutable special text tokens and tokens requiring this are not supported.

EJBCA End Entity for EJBCA CA Access

Keyfactor Command supports synchronization of certificates and certificate enrollment from EJBCA certificate authorities by configuring a client certificate issued from the EJBCA CA on the CA record in the Management Portal. This client certificate needs to be associated with an end entity in EJBCA that can be assigned sufficient permissions to perform all necessary CA tasks from Keyfactor Command.

Explicit Credentials for Microsoft CA Access

Keyfactor Command supports synchronization of certificates and certificate enrollment from Microsoft certificate authorities in remote forests (forests other than the forest in which Keyfactor Command is installed which are not in a two-way trust with the Keyfactor Command forest) by configuring a service account from the forest in which the CA resides on the CA record in the Management Portal. All communication to retrieve existing certificates, enroll for new certificates, revoke certificates, and recover certificate keys from the remote CA is done in the context of this service account. Explicit credentials for remote CA access is configured in the Keyfactor Command Management Portal after installation is complete rather than in the configuration wizard.

You may need additional service accounts to support the use of Keyfactor Command orchestrators and/or gateways in your environment. Please see:

- Create Service Accounts for the Universal Orchestrator on page 2884
- Create a Service Account for the Keyfactor Bash Orchestrator on page 2994

- Create Service Accounts for the Java Agent on page 2969
- The installation guide for each gateway.

The service account(s) need to be created in Active Directory prior to installation of the Keyfactor Command software, and the person installing the Keyfactor Command software needs to know the service account(s) domain, username and password. The same service account may be used for multiple roles, if desired. For example, you might have one service account for orchestrators, another for gateways, and a third for all server roles.

Table 845: Typical Service Accounts

Account	Uses
Keyfactor Command Service Account	Keyfactor Command Service, Keyfactor Command Management Portal (Application Pool), Keyfactor API, Keyfactor Command Logi Report Access
Keyfactor Orchestrator Service Account	Keyfactor Orchestrator access to Keyfactor Command Server and Keyfactor Orchestrator on-machine operations, where applicable

4.4.2.6 Create Groups to Control Access to Keyfactor Command Features

Keyfactor Command uses groups to control access to the various Keyfactor Command features. The Keyfactor Command Management Portal supports multiple groups with different levels of access to the portal. During the installation, at least one group or user must be entered to grant full administrative access to the portal. After installation, additional groups can be configured through the Keyfactor Command Management Portal to grant more limited access to the portal.

Important: The built-in Active Directory groups Domain Admins and Enterprise Admins cannot be used directly to grant access to the Management Portal due to how these groups function within Windows. You can create a custom Active Directory group, reference that group in the Management Portal, and add the built-in Domain Admins or Enterprise Admins group to that custom group, if desired.

Groups that you may find it useful to identify or add following the initial installation include:

Keyfactor Command Enrollment

Users who are a member of this group or groups may use PFX and/or CSR enrollment through the Keyfactor Command Management Portal. Access control for enrollment is configured in the Keyfactor Command Management Portal after installation is complete.

Keyfactor Command My SSH Key

Users who are a member of this group or groups may acquire SSH keys through the Keyfactor Command My SSH Key portal. Access control for the My SSH Key portal is configured in the Keyfactor Command Management Portal after installation is complete.

Keyfactor Java Agents

Service accounts that are a member of this group are allowed to auto-register as Keyfactor Java Agents in Keyfactor Command, if auto-registration is configured, providing for more hands-free management of Java and PEM certificate stores. This group is not required if auto-registration with user validation will not be used.

Keyfactor Bash Orchestrators

Service accounts that are a member of this group are allowed to auto-register as Bash orchestrators in Keyfactor Command, if auto-registration is configured, providing for more hands-free management of Bash orchestrators. This group is not required if auto-registration with user validation will not be used.

Keyfactor Mac Auto-Enrollment Users

Users who are members of this group are allowed to auto-register for Mac auto-enrollment in Keyfactor Command, if auto-registration is configured, providing for more hands-free management of Mac auto-enrollment. The same group may be used to grant users permissions on the template that will be used for Mac auto-enrollment. This group is not required if auto-registration with user validation will not be used and a different group will be used to grant permission on the template.

Keyfactor Universal Orchestrators

Service accounts that are a member of this group are allowed to auto-register as Keyfactor Universal Orchestrators in Keyfactor Command, if auto-registration is configured, providing for more hands-free management of certificate stores managed by the Keyfactor Universal Orchestrator. This group is not required if auto-registration with user validation will not be used.

Note: The same group may be used for multiple roles. Existing groups may be used. For example, if all employees of your organization are members of the Active Directory Domain Users group and you wish to allow all employees to acquire SSH keys, you may use the Domain Users group for the Keyfactor Command My SSH Key function.

Tip: If you're using Active Directory as an identity provider and wish to grant access in the Management Portal to users from trusted Active Directory forests, create a domain local group in the Active Directory domain in which Keyfactor Command is installed, put the cross-forest users and groups in this local group and grant access in Keyfactor Command to this domain local group.

4.4.2.7 Configure Certificate Chain Trusts for CAs

The Keyfactor Command server needs to trust the chain certificates for all the CAs you will reference within Keyfactor Command in order for all operations to complete successfully. In many environments, root and intermediate trusts for domain-joined Microsoft CAs are pushed out automatically.

If this is not the case in your environment or if you are using non-domain-joined CAs (e.g. EJBCA CAs), you will need to configure these chain trusts on the Keyfactor Command server manually.

The certificate for each root CA must be installed in the Trusted Root Certification Authorities store under Local Computer on the Keyfactor Command server. If your public key infrastructure (PKI) also has issuing CAs, the issuing CA certificates must be installed in the Intermediate Certification Authorities store under Local Computer on the Keyfactor Command server.

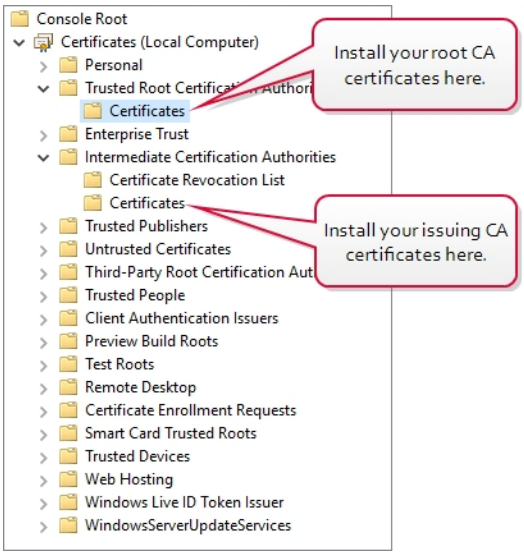


Figure 495: Install CA Chain Certificates on the Keyfactor Command Server

4.4.2.8 Hostname Identification and Resolution

Prior to the installation of Keyfactor Command, you need to determine the DNS alias(es) by which the Keyfactor Command roles will be accessed, if any, and configure them in your hostname resolution solution so that they will be resolvable prior to installation. For example, if you're licensed for SSH key management and wish to publish the My SSH Key portal externally to support SSH key acquisition by users outside the company firewall, you will probably wish to reference the server by a DNS alias rather than its actual hostname. For example, you may wish to use keyfactor.keyexample.com rather than websrvr23.keyexample.local. This is particularly significant if you will be using redundant servers with load balancing.

For DNS aliases used internally in environments using Active Directory as an identity provider, you will need to consider whether the servers to be accessed will be authenticated using Kerberos authentication. Out of the box, the Keyfactor Command Management Portal uses integrated Windows authentication and will default to Kerberos authentication in most environments. Although some features of the Keyfactor Command Management Portal may support NTLM authentication in some environments, the dashboard and enrollment functions do not support NTLM. If you will be using Kerberos authentication, your DNS aliases need to be configured as "A" records rather than CNAME records because Kerberos does not function well with CNAME records under Microsoft IIS.

The roles for which you need hostnames during the Keyfactor Command installation are:

SQL Server

For a small environment you may choose to use the server's actual name. If you plan to use SQL clustering, you will need an alias that represents the cluster. Using an alias for the SQL server allows for database portability in the future.

Email

During the Keyfactor Command installation you configure the email server that will be used to send email notifications.

Keyfactor Command Management Portal

This is the primary management server and may hold all Keyfactor Command roles in a small implementation.

Keyfactor Command Logi Dashboard and Reports

This hostname must match the hostname entered for the Management Portal.

Keyfactor Command Orchestrators API

This hostname is only required if your Keyfactor Command license includes orchestrator functionality. If all Keyfactor Command roles are combined on one server, this will be the same hostname as used for the Keyfactor Command Management Portal.

Keyfactor Command Keyfactor API

This hostname must match the hostname entered for the Management Portal unless you are installing a secondary instance of the Keyfactor API.

Centralized Logging Solution

This hostname is only required if you choose to enable the option to copy Keyfactor Command audit logs entries in real time, as they are generated, to a separate server for collection and analysis by a centralized logging solution (e.g. rsyslog, Logstash).

Prior to beginning the Keyfactor Command installation, ensure that the selected hostnames resolve successfully.

4.4.2.9 Firewall Considerations

Keyfactor Command needs to be able to communicate internally between the various Keyfactor Command components installed on different servers, if applicable, and to the SQL server, certificate authorities, centralized logging server (if applicable), your identity provider. If there are any firewalls in the environment that control internal traffic, these may need to be updated to allow the appropriate level of communication. Table 846: Protocols Keyfactor Command Uses for Communication shows each Keyfactor Command component and the protocols they use to communicate. In

environments using Active Directory as an identity provider, all Keyfactor Command components require a healthy Active Directory environment with the ability to use Kerberos, LDAP, and DNS.

Table 846: Protocols Keyfactor Command Uses for Communication

Keyfactor Command Component	Protocols and Ports	Target
Keyfactor Command Management Portal	HTTP/HTTPS (TCP 80/443)	Client browser (e.g. Microsoft Edge)
Keyfactor Command Management Portal	HTTP/HTTPS (TCP 80/443)	Certificate revocation list (CRL) distribution points
Keyfactor Command Management Portal	HTTP/HTTPS (TCP 80/443)	EJBCA Certificate Authorities
Keyfactor Command Management Portal	RPC/DCOM (TCP 135 plus random high ports typically in the range 49152 – 65535)	Microsoft Certificate Authorities
Keyfactor Command Management Portal	RPC/DCOM (TCP 135 plus random high ports typically in the range 49152 – 65535)	Keyfactor vendor gateways to cloud CAs (e.g. Entrust, Symantec)
Keyfactor Command Management Portal	MS SQL (default TCP 1433)	SQL Server
Keyfactor Command Management Portal	Varies depending on the implemented solution (TCP 514 for rsyslog, TCP 5000 for Logstash are some standard defaults)	Centralized logging solution
Keyfactor Command	Active Directory (TCP/UDP 389)	Microsoft Active Directory queries
Keyfactor Command SSH Management	Active Directory Web Services (TCP 9389)	Microsoft Active Directory for group membership enumeration
All Orchestrators and Agents	HTTP/HTTPS (TCP 80/443)	Keyfactor Command Orchestrator API endpoint
Keyfactor Universal Orchestrator with Extension Relying on PowerShell Remoting and WinRM	PowerShell Remoting (default TCP 5985 and 5986)	Windows Servers to which certificate files will be distributed

Keyfactor Command Component	Protocols and Ports	Target
(IIS and Remote File Extensions)		
Keyfactor Universal Orchestrator (SSL Endpoint Management)	Any configured for scanning	The SSL endpoint being scanned by the SSL discovery or monitoring job
Keyfactor Universal Orchestrator with Extension Relying on HTTP/HTTPS (F5 and Citrix NetScaler Certificate Store Management)	HTTP/HTTPS (TCP 80/443)	F5 or NetScaler Devices
Keyfactor Universal Orchestrator (Remote Certificate Authority)	RPC/DCOM (TCP 135 plus random high ports typically in the range 49152 – 65535)	Microsoft Certificate Authorities
Keyfactor Bash Orchestrator	SSH (TCP 22 by default)	Remote control targets for SSH management
Keyfactor Gateways to Cloud CAs	HTTP/HTTPS (TCP 80/443)	Cloud providers (e.g. Entrust, Symantec)
Keyfactor Cloud Gateway	Active Directory Web Services (TCP 9389)	Microsoft Active Directory for group membership enumeration

4.4.2.10 Acquire a Public Key Certificate for the Keyfactor Command Server

Keyfactor recommends using HTTPS to secure the channel between clients and the Keyfactor Command server(s). This requires at least one SSL certificate. You will need an SSL certificate or certificates for each of the hostnames you have identified (see Hostname Identification and Resolution on page 2764).

Acquire the certificate(s) using the Fully Qualified Domain Name (FQDN) of the server or alias used for the Keyfactor Command server(s). For example:

```
keyfactor.keyexample.com
```

The certificate(s) may be installed on the Keyfactor Command server(s) prior to installation of the Keyfactor Command software or may be installed at the time of Keyfactor Command installation. See Configure SSL for the Default Web Site on the Keyfactor Command Server on page 2774 for more information.

If installed ahead of time, the certificate(s) should be placed in the Personal Certificate store of the Local Computer using the Certificates MMC Snap-In.

4.4.2.11 Install IIS and .NET on the Keyfactor Command Server

Internet Information Services (IIS) and .NET 4.7.2 or greater must be installed on the Keyfactor Command server(s) prior to installation of the Keyfactor Command software.

IIS is a standard Windows role added through the Windows Server Manager tool and .NET is a standard Windows feature added through the Windows Server Manager tool. You may need to update to .NET 4.7.2 or greater with a downloadable update package or through Windows update.

> ⚠ **Important:** IIS needs to be configured to allow requests using the HTTP verbs DELETE, GET, POST and PUT to reach the Default Web Site (or other web site if you choose to install to an alternate web site). These are enabled by default. To check whether any of these have been disabled, open the IIS Management console, drill down to highlight the Default Web Site, double-click **Request Filtering** in the center pane, and review the information on the **HTTP Verbs** tab.

To verify the version of .NET installed, either:

1. Open the Registry Editor:

   ```
   regedit
   ```

2. Navigate to:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full
   ```

3. Validate that the **Release** attribute value indicates a version of .NET Framework that is 4.7.2 or higher is installed, as shown in Table 847: .NET Framework Release Values.

Or:

1. Open a command prompt or PowerShell window and type the following command:

   ```
   reg query "HKLM\Software\Microsoft\NET Framework Setup\NDP\v4\Full"
   ```

2. Validate that the **Release** attribute value indicates a version of .NET Framework that is 4.7.2 or higher is installed, as shown in Table 847: .NET Framework Release Values.

Table 847: .NET Framework Release Values

.NET Framework	Release Value (Decimal)	**Release Value (Hexadecimal)**
.NET Framework 4.6.2	394802 or 394806	60632 or 60636

.NET Framework	Release Value (Decimal)	**Release Value (Hexadecimal)**
.NET Framework 4.7	460805	70805
.NET Framework 4.7.1	461308 or 461310	709FC or 709FE
.NET Framework 4.7.2	461808 or 461814	70BF0 or 70BF6
.NET Framework 4.8	528040, 528049, 528372, or 528449	8oEA8, 80EB1, 80FF4, 81041

Installing IIS and ASP.NET on Windows Server 2019 and 2022

The following figures show the components of IIS and ASP.NET necessary to support Keyfactor Command on Windows Server 2019 and 2022. Your Keyfactor Command server may have additional roles or features installed that are not shown in these figures.

Important: Do not install the IIS *WebDAV Publishing* feature. Keyfactor Command will not operate correctly if this feature is installed.

Keyfactor Command makes use of the Active Directory tools for PowerShell to do group membership queries in Active Directory in some functions (e.g. when using a group to create a mapping between a Linux logon for SSH and one or more SSH keys). The *Active Directory module for Windows PowerShell* is installed as a feature as part of the *Remote Server Administrator Tools*.

Note that it is possible to install IIS and the necessary features using PowerShell rather than the below-referenced GUI-based installation method. The correct PowerShell command for this is:

```
Install-WindowsFeature Web-Server, Web-Asp-Net45, Web-Default-Doc, Web-Dir-Browsing, Web-Http-Errors,
Web-Static-Content, Web-Http-Logging, Web-Stat-Compression, Web-Filtering, Web-Basic-Auth, Web-
Windows-Auth, Web-Net-Ext45, Web-ISAPI-Ext, Web-ISAPI-Filter, Web-Mgmt-Console, RSAT-AD-PowerShell
```

Tip: To check and see if all the required roles and features have been installed, use Get-WindowsFeature with the same list of roles and features like so:

```
Get-WindowsFeature Web-Server, Web-Asp-Net45, Web-Default-Doc, Web-Dir-Browsing,
Web-Http-Errors, Web-Static-Content, Web-Http-Logging, Web-Stat-Compression, Web-
Filtering, Web-Basic-Auth, Web-Windows-Auth, Web-Net-Ext45, Web-ISAPI-Ext, Web-
ISAPI-Filter, Web-Mgmt-Console, RSAT-AD-PowerShell
```

Output from this command will look something like the following, which shows some required features installed and some missing. Make sure all roles and features in the query output are marked *Installed* before continuing.

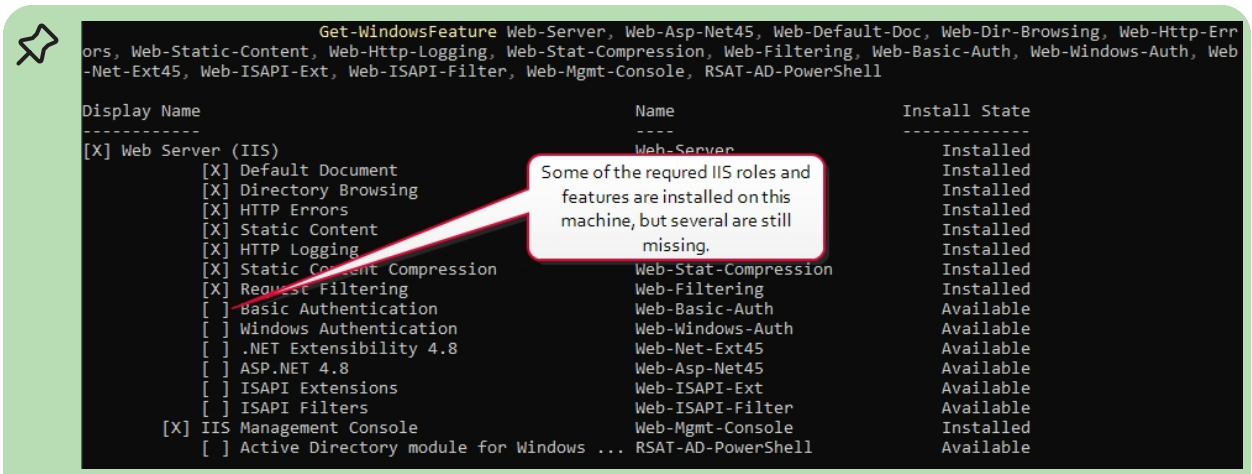


Figure 496: Use Get-WindowsFeature to Determine if All Required Roles and Features are Installed

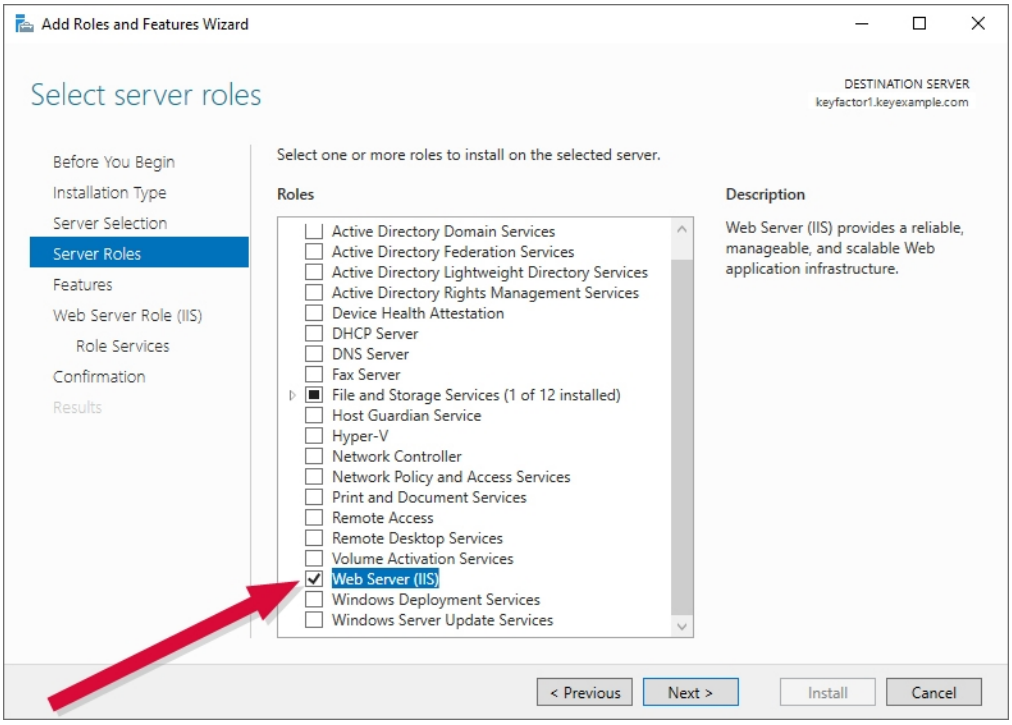


Figure 497: Web Server Role

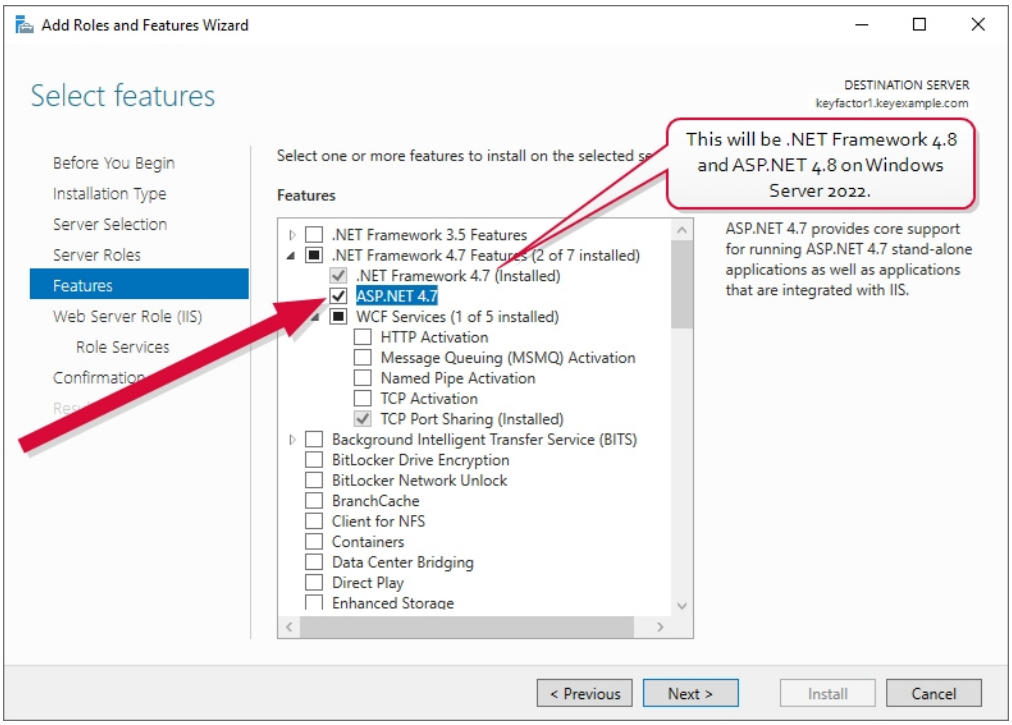


Figure 498: .NET 4.7 Feature

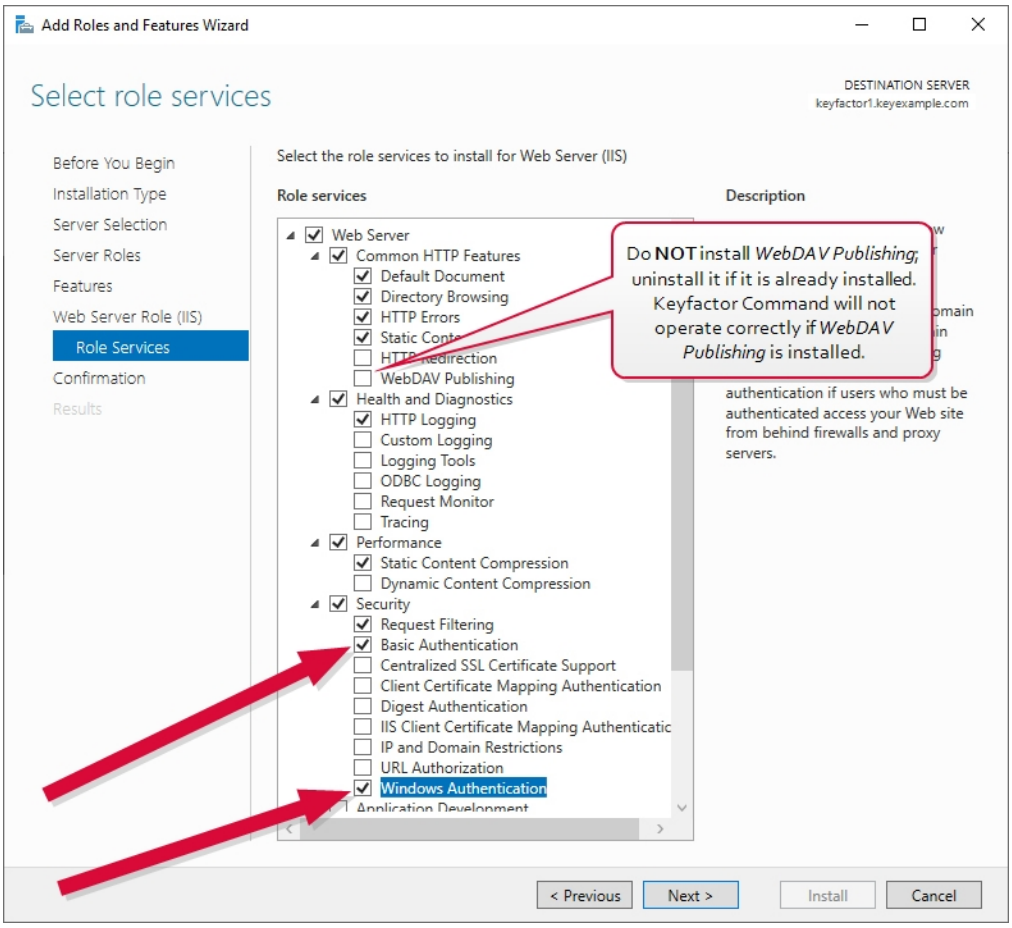


Figure 499: Role Services Page One

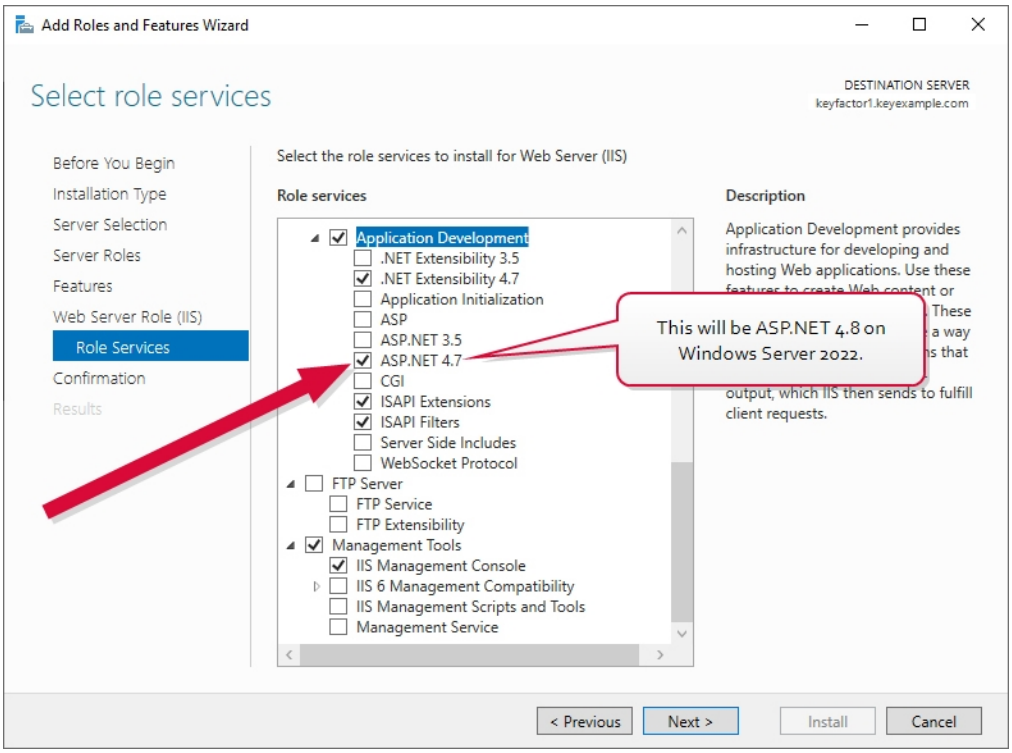


Figure 500: Role Services Page Two

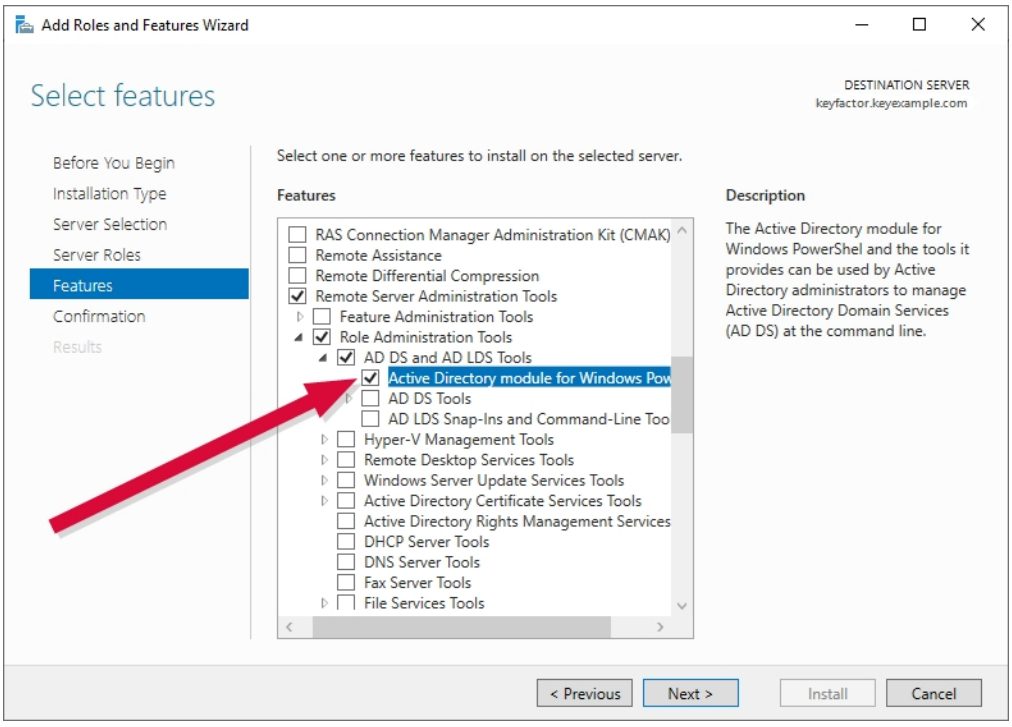


Figure 501: Active Directory Module for Windows PowerShell

4.4.2.12 Configure SSL for the Default Web Site on the Keyfactor Command Server

Once you have acquired an SSL certificate for Keyfactor Command and installed IIS, you can open the IIS Management Console and associate the certificate with the Default Web Site. You can do this either before or after installing Keyfactor Command.

To import your SSL certificate and associate it with the Default Web Site:

1. Open the IIS Manager MMC snap-in.
2. Navigate to the connection for the current host. (The top level in IIS.)
3. On the current host Home page, open (double-click) Server Certificates. If your SSL certificate already appears in this list, you can skip steps 4-7.
4. On the Server Certificates page, select Import... under Actions.
5. In the Certificate file (.pfx) field, choose the browse option and navigate to the .pfx or .p12 file containing your certificate.
6. Enter the password for your certificate, select the Personal Certificate Store, check the Allow this certificate to be exported box if desired, and click **OK**.
7. Your certificate should now appear in the list of Server Certificates. Confirm that the Issued To column shows your certificate name correctly (e.g. keyfactor.keyexample.com).
8. Navigate to the Default Web Site and on the Default Web Site Home page, select Bindings... under Actions.
9. In the Site Bindings dialog, highlight the https entry if it exists and choose Edit. If an https entry does not exist, click **Add**.
10. In the Edit Site Bindings dialog, select https in the Type dropdown (this will already be selected and grayed out if you selected Edit in the previous step), select the certificate you just imported in the SSL certificate dropdown box, and click **OK**.

Note that these instructions assume that your SSL certificate has been provided in PKCS12 format file. If you are requesting a certificate directly from an on-premise CA through IIS or are generating a CSR through this IIS installation to submit to a CA, the configuration steps will be different.

4.4.2.13 Configure the Keyfactor Command Server to Require SSL

For best security practice, the Keyfactor Command web site should be configured to require SSL for all access. To do this:

1. Open the IIS Manager MMC snap-in.
2. Navigate to the Default Web Site.
3. Under the Default Web Site Home, select **SSL Settings**.
4. On the SSL Settings page, check the **Require SSL** box and, under Actions, click **Apply**.

Important: The Keyfactor Command web application is not configured to support HTTP Strict Transport Security (HSTS) by default. HSTS is a web security policy mechanism that helps to protect websites against protocol downgrade attacks and cookie hijacking. An application enables HSTS by returning an HTTP response header that instructs users' browsers to only interact with the site using secure transport methods. HSTS is supported by all modern browsers. To accommodate this, configure the server to always send the *Strict-Transport-Security* HTTP header on HTTPS connections:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

The max-age parameter is given in seconds; the value shown above is equivalent to one year.

Instructions for implementation of HSTS are beyond the scope of this guide.

4.4.2.14 Prepare for External Log Shipping over TLS (Optional)

Keyfactor Command offers the option to copy audit logs in real time to a separate server for collection and analysis with a centralized logging solution (e.g. rsyslog, Splunk, Elastic Stack). This can be done either over standard UDP/TCP connections, or you can opt to secure the connection to the backend log collection solution using TLS. This requires a backend solution that supports receiving logs over TLS and, typically, a client certificate on the Keyfactor Command server and a server certificate on the backend server.

The following instructions cover using rsyslog on the backend and will differ if you are using an alternative log collection solution.

Acquire the client certificate(s) using the Fully Qualified Domain Name (FQDN) of the server or alias used for the Keyfactor Command server(s) (see Hostname Identification and Resolution on page 2764). For example:

```
keyfactor.keyexample.com
```

The certificate must be installed on the Keyfactor Command server(s) prior to installation of the Keyfactor Command software.

To acquire a client certificate for use in log shipping using a Microsoft CA, first create or identify a template that has an extended key usage (EKU) of *Client Authentication* and make it available for enrollment on a CA to which the Keyfactor Command server has access with enrollment permissions for the Keyfactor Command server. If desired, you can use a template that has both the *Client Authentication* and *Server Authentication* EKUs and use it for certificates on both sides of the communication. Start by copying a computer template if you want to enroll for the certificate using the Microsoft MMC as described below and without needing to set the private key of the certificate as exportable.

To enroll for a client certificate using the MMC:

1. On the Keyfactor Command server, do one of following:

    - Using the GUI:
        a. Open an empty instance of the Microsoft Management Console (MMC).
        b. Choose **File->Add/Remove Snap-in...**.
        c. In the *Available snap-ins* column, highlight **Certificates** and click **Add**.
        d. In the Certificates snap-in popup, choose the radio button for Computer account, click **Next**, accept the default of Local computer, and click **Finish**.
        e. Click **OK** to close the Add or Remove Snap-ins dialog.
    - Using the command line:
        a. Open a command prompt using the "Run as administrator" option.
        b. Within the command prompt type the following to open the certificates MMC:

            ```
            certlm.msc
            ```

2. Drill down to the Personal folder under **Certificates** for the Local Computer, right-click, and choose **All Tasks->Request New Certificate...**.
3. Follow the certificate enrollment wizard, selecting the template you created or identified for use for this purpose, and providing any required information, being sure to set the CN to the FQDN of the Keyfactor Command server.

Tip: If you have an existing Keyfactor Command and wish to enroll through Keyfactor Command, you can request the certificate using the PFX enrollment option and either push it out to the Keyfactor Command local machine store using an IIS personal certificate store managed with a Keyfactor Universal Orchestrator installed on Windows with the IIS custom extension (see Installing Custom-Built Extensions on page 2940) as part of the enrollment process or import it to the certificate store using the PFX generated from Keyfactor Command.

Note: The Keyfactor Command server needs to be configured to trust the CA that issued the certificate to the rsyslog server. If you have opted to acquire certificates from a CA for which a root trust is not already configured on the Keyfactor Command server, this will need to be configured.

To acquire a server certificate for use in log shipping using a Microsoft CA, first create or identify a template that has an extended key usage (EKU) of *Server Authentication* and make it available for enrollment on a CA to which the server from which you are requesting the certificate has access with enrollment permissions for the server from which you are requesting the certificate.

To enroll for a server certificate using the MMC:

1. On a Windows server with appropriate enrollment access, do one of following:

    - Using the GUI:
        a. Open an empty instance of the Microsoft Management Console (MMC).

        b. Choose **File->Add/Remove Snap-in...**.

        c. In the *Available snap-ins* column, highlight **Certificates** and click **Add**.

        d. In the Certificates snap-in popup, choose the radio button for Computer account, click **Next**, accept the default of Local computer, and click **Finish**.

        e. Click **OK** to close the Add or Remove Snap-ins dialog.

    - Using the command line:
        a. Open a command prompt using the "Run as administrator" option.

        b. Within the command prompt type the following to open the certificates MMC:

            certlm.msc

2. Drill down to Certificates in the Personal folder under **Certificates** for the Local Computer and locate your newly created certificate. Right-click on the certificate and choose **All Tasks->Export...**.

3. Follow the certificate export wizard, being sure to answer **Yes, export the private key** and choosing the option to **Include all certificates in the certificate path if possible**. Set a password to secure the exported private key.

4. In the MMC, drill down to the Personal folder under **Certificates** for the Local Computer, right-click, and choose **All Tasks->Request New Certificate...**.

5. Securely copy the resulting PFX file to your rsyslog server and place it in a temporary working directory.

6. Use openssl to break the PFX file apart into separate certificate and key files and remove the encryption on the key file (rsyslog does not provide a method for providing the password necessary to use the encrypted file) as follows:

    a. Execute the following command to pull the key out of the PFX file (provide the input password for the PFX when prompted and the output password for the PEM key file when prompted):

        ```
        openssl pkcs12 -in mycertfile.pfx -nocerts -out mykey.pem
        ```

    b. Execute the following command to pull the certificate out of the PFX file (provide the input password for the PFX when prompted):

        ```
        openssl pkcs12 -in mycertfile.pfx -clcerts -nokeys -out mycert.pem
        ```

c. Execute the following command to pull the chain certificate(s) out of the PFX file (provide the input password for the PFX when prompted):

```
openssl pkcs12 -in mycertfile.pfx -cacerts -nokeys -chain -out cacerts.pem
```

d. Execute the following command to remove the encryption from the key so that a password will not be required when accessing the key file (provide the PEM key password you set in the first step):

```
openssl rsa -in mykey.pem -out mynewkey.key
```

7. Identify a secure location on the rsyslog server to store the certificates and key file (e.g. /etc/tls/certs and /etc/tls/private) and copy the certificates and key to these locations, setting appropriately secure permissions on the files. The key needs to be readable by the rsyslog daemon.

Tip: If you have an existing Keyfactor Command and wish to enroll through Keyfactor Command, you can request the certificate using the PFX enrollment option, opt to download it as a ZIP PEM, copy the zip file to the rsyslog server, unzip, and distribute the files as described in the final step, above.

Configuration of rsyslog for TLS support may vary depending on your needs. In addition to the standard rsyslog package for your Linux server, you will need GNU TLS packages to support TLS communication. For example, for Ubuntu the required packages are:

```
rsyslog
rsyslog-gnutls
gnutls-bin
```

The following is an example rsyslog.conf file configured for TLS support:

```
# /etc/rsyslog.conf configuration file for rsyslog
#
# For more information install rsyslog-doc and see
# /usr/share/doc/rsyslog-doc/html/configuration/index.html
#
# Default logging rules can be found in /etc/rsyslog.d/50-default.conf
#################
#### MODULES ####
#################

module(load="imuxsock") # provides support for local system logging
#module(load="immark")  # provides --MARK-- message capability

# provides UDP syslog reception
```

```
#module(load="imudp")
#input(type="imudp" port="514")

# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")

# provides kernel logging support and enable non-kernel klog messages
module(load="imklog" permitnonkernelfacility="on")

###########################
#### GLOBAL DIRECTIVES ####
###########################

#
# Use traditional timestamp format.
# To enable high precision timestamps, comment out the following line.
#
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# Filter duplicated messages - It can be helpful to set this 'off' during initial Keyfactor Command
testing
$RepeatedMsgReduction off

#
# Set the default permissions for all log files.
#
$FileOwner syslog
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022
$PrivDropToUser syslog
$PrivDropToGroup syslog

#
# Where to place spool and state files
#
$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
#
$IncludeConfig /etc/rsyslog.d/*.conf
```

```
#
# Configuration for TLS
#
$DefaultNetstreamDriver gtls

$DefaultNetstreamDriverCAFile /etc/tls/certs/cacerts.pem
$DefaultNetstreamDriverCertFile /etc/tls/certs/mycert.pem
$DefaultNetstreamDriverKeyFile /etc/tls/private/mynewkey.key

$ModLoad imtcp

$InputTCPServerKeepAlive on
$InputTCPServerStreamDriverAuthMode anon
$InputTCPServerStreamDriverMode 1 # run driver in TLS-only mode

$ActionSendStreamDriverAuthMode x509/name
$ActionSendStreamDriverMode 1 # run driver in TLS-only mode

$InputTCPServerRun 10514
```

A filter similar to the following can be used to redirect all Keyfactor Command related traffic to a particular file:

```
:syslogtag, isequal, "Keyfactor" /var/log/keyfactor/audit.log
```

4.4.3 Installing

The following installation instructions cover installing all Keyfactor Command server components on a single server performing all Keyfactor Command roles. You may choose to separate the roles onto different servers. If you do, the installation process will vary from the described process.

4.4.3.1 Install the Keyfactor Command Components on the Keyfactor Command Server(s)

Before you begin the installation, make sure that you have reviewed the system requirements (see System Requirements on page 2702), completed the prerequisites (see Planning & Preparing on page 2704), and have your Keyfactor Command license file ready to upload during the configuration.

The following installation steps show all possible Keyfactor Command features enabled. Your Keyfactor Command license may not cover all Keyfactor Command features. If it does not, unlicensed features will not be shown in the configuration wizard. You may skip those configuration steps.

To begin the Keyfactor Command installation, execute the KeyfactorPlatform.msi file from the Keyfactor Command installation media and install as follows.

1. On the first installation page, click **Next** to begin the setup wizard.

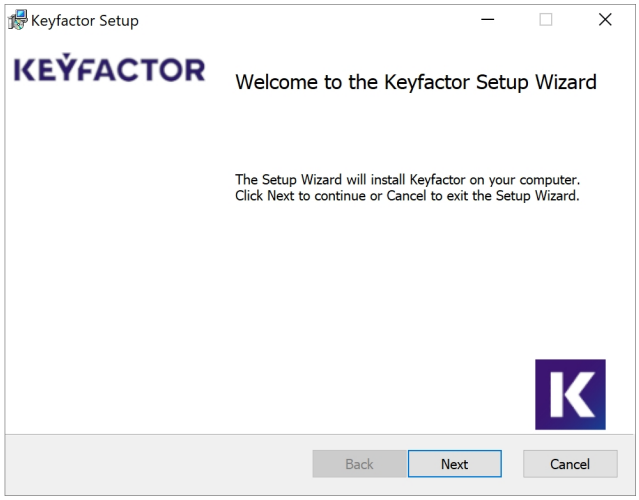


Figure 502: Install: Begin Setup Wizard

2. On the next page, read and accept the license agreement and click **Next**. Click **Print** to review a printed copy if desired.

3. On the next page, select the components to install. For a server with the default roles collocated, leave the default options and click **Next** to continue. If desired, you can highlight Keyfactor Command and click **Browse** to select an alternate installation location for the files. The default installation location is:

```
C:\Program Files\Keyfactor\Keyfactor Platform\
```

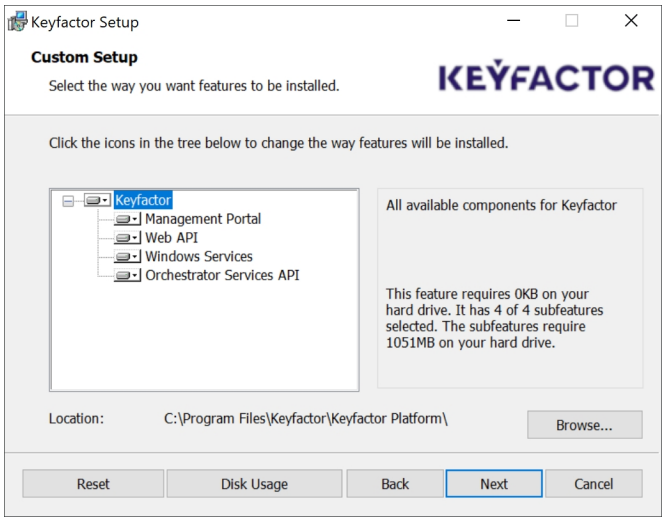


Figure 503: Install: Select Components

Tip: Refer to Keyfactor Command Server(s) on page 2756 for a description of these components.

Table 848: Available components for Keyfactor.

Component	Description
Management Portal	**Mandatory**. Web-based management console for configuring all aspects of Keyfactor. The Keyfactor API will be installed with this component.
Windows Services	**Mandatory**. Includes the timer Windows service to manage timed events, such as CA Sync, PKI monitoring and system maintenance.
Web API	Optional. The Keyfactor API component. This allows the Keyfactor API for external use to be installed on a separate server from the Management Portal, if desired.
Orchestrator Services API	Optional. Not required if neither agents nor orchestrators will be utilized by Keyfactor Command. Web based orchestrator services API.

4. On the next screen, click **Install**.

5. On the final installation wizard page, leave the *Launch the Configuration Wizard now* box selected and click **Finish**. The configuration wizard should start automatically. This can take several seconds.

6. On the Keyfactor Command Database Configuration page, enter the name, IP address, or fully qualified domain name (FQDN) of your SQL server and select a Credential Type of either

Windows or **SQL**.

Important: Keyfactor Command uses an encrypted channel to connect to the SQL server by default, which requires configuration of an SSL certificate on the SQL server (see Using SSL to Connect to SQL Server on page 2746). The name or IP address you enter here for your SQL server must be available as a SAN in this certificate unless you have disabled the encrypted connection for Keyfactor Command (see Configurable SQL Connection Strings on page 2750).

- If you select **Windows** as the Credential Type for connecting to SQL, click the **Connect** button.

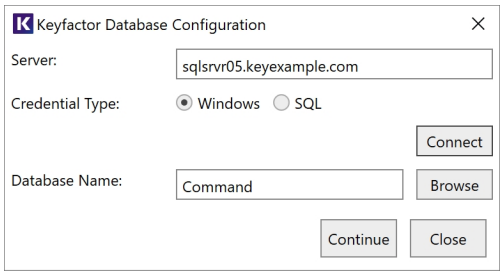


Figure 504: Windows Authentication

- If you select **SQL** as the Credential Type for connecting to SQL, the window will expand to include fields to enter a SQL username and password. Enter a username and password to authenticate to SQL, and click the **Connect** button.

  Note: The password must not contain single or double quotes. An error will be shown if single or double quotes are used in the password.

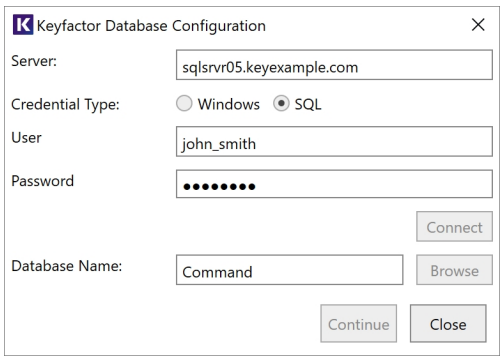


Figure 505: SQL Authentication

Note: For the permissions required for this user, see Grant Permissions in SQL on page 2744.

Note: Keyfactor Command supports configuration of a base SQL connection template that is used for all connections Keyfactor Command makes to SQL. For more information, see Configurable SQL Connection Strings on page 2750.

Note: Your SQL server must be configured to support mixed mode authentication in order to use the SQL option.

7. After the **Connect** button is clicked, the database name field will be activated. You can either enter the name of the desired database—for either a new or existing database—or click **Browse** to scroll through a list of existing databases.

   Note: On subsequent runs of the configuration wizard, the database name field will be pre-populated with the database name used on the last completed run. Any change to the server connection fields (server name, authentication type, etc.) will require the Connect button to be used again to unlock the database name field and the Continue button.

8. Click the **Continue** button. You will receive a confirmation dialog if any changes will be made to the database at this stage.

   Note: If any of the following situations occurs, you will receive a message:

   - The selected database does not exist and will be created.
   - The selected database is empty and not associated with Keyfactor Command; it will be populated with the Keyfactor Command schema.
   - The selected database does not match the current product schema and will be upgraded.
   - The selected database is not empty and is not associated with Keyfactor Command.
   - The user does not have access to the database.
   - An SSL certificate is not correctly configured on the SQL server.

9. On the Keyfactor Command Encryption Warning page, read and understand the warning. Make note of the referenced documents to provide to your SQL team. Take advantage of the option to make a backup of the Database Master Key (DMK) by entering a path to a directory on your SQL server along with a filename for the backup file and a password to encrypt the file and clicking **Backup**. The user running the Keyfactor Command installer must have write permissions to this directory. Click **Continue**.

   Important: Keyfactor Command uses Microsoft SQL Server encryption to protect security sensitive data, including service account credentials. Backup of the SQL server Database Master Key (DMK) is of critical importance in database backup and recovery

operations. The backup file of the DMK and the password should be stored in a safe, well-documented location. Without the file and password created with this process, some data that is encrypted within the Keyfactor Command database will be unrecoverable in a disaster recovery scenario. For more information, see SQL Encryption Key Backup on page 821.

If you choose to install Keyfactor Command in the default location, the referenced documents can later be found here:

```
C:\Program Files\Keyfactor\Keyfactor Platform\Configuration\DMKBackup.docx
C:\Program Files\Keyfactor\Keyfactor Platform\Configuration\DMKRestore.docx
```

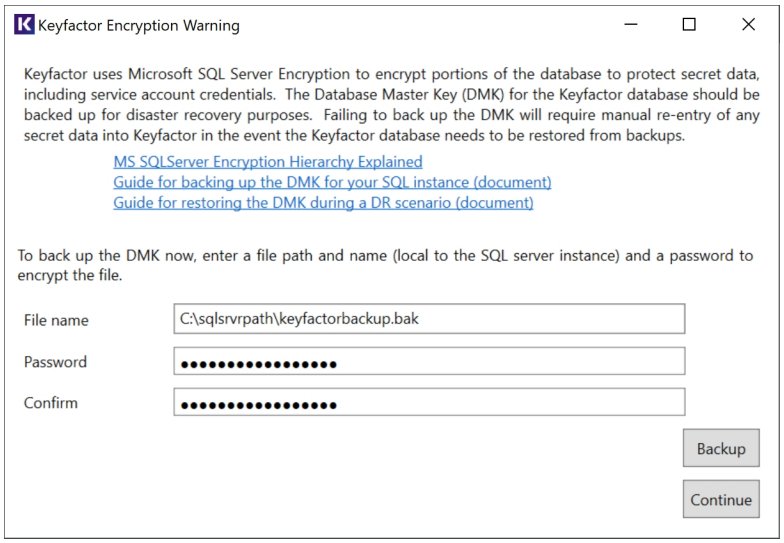


Figure 506: Configure: Backup Database Master Key

10. On the Keyfactor Command License upload page, click **Upload** and browse to locate the license file provided to you by Keyfactor. This file should have the extension CMSLICENSE. Once the uploaded license shows as valid, click **Continue**.

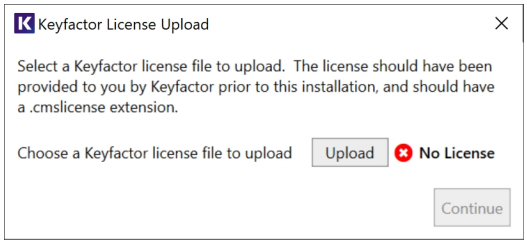


Figure 507: Configure: Upload License

11. In the Keyfactor Command configuration wizard, you can choose to upload a configuration file to populate the fields. You may have a file saved from a previous run of the configuration wizard or you may be provided one by Keyfactor. To upload a file, in the configuration wizard, click **File** at the top of the wizard and choose **Open Data File**. Browse to locate the configuration file. Configuration files have an extension of .cmscfg. The file may be protected with a password. If it is, you will need to provide this password to open the file. Continue with the remainder of the steps, reviewing the tabs to assure that the data is complete and correct.

    Note: If you open a configuration file that contains configuration information for an identity provider other than Active Directory but does not set OAuth enabled to true, the additional identity provider information will not be loaded.

    Figure 508: Configure: Open Data File

    Note: At the bottom of the configuration wizard, if the database server name is longer than will fit in the provided window, it will be truncated and an ellipsis will be added.

12. **Application Pools Tab**

    A separate application pool is required for each virtual directory that will be created for Keyfactor Command in IIS. If you've chosen to install all the Keyfactor Command components, this will be five application pools for the virtual directories with the following names, by default:

    - KeyfactorAgents (Keyfactor Command agent and orchestrator service endpoint)
    - KeyfactorAnalysis (Keyfactor Command dashboard and reporting)
    - KeyfactorAPI (Keyfactor API)
    - KeyfactorProxy (Authentication for an identity provider other than Active Directory)
    - KeyfactorPortal (Keyfactor Command Management Portal)

    On the Application Pools tab of the configuration wizard, click **Add**, change the default application pool name, if desired, and enter the user name (DOMAIN\username format) and password of the Active Directory service account under which the application pool will run. You may use the people picker button ( ) to browse for the account. Click the verify button ( ) to confirm that the username and password entered are valid. Assuming the verification completes successfully, click **Save**.

    Tip: The same service account may be used for all application pools.

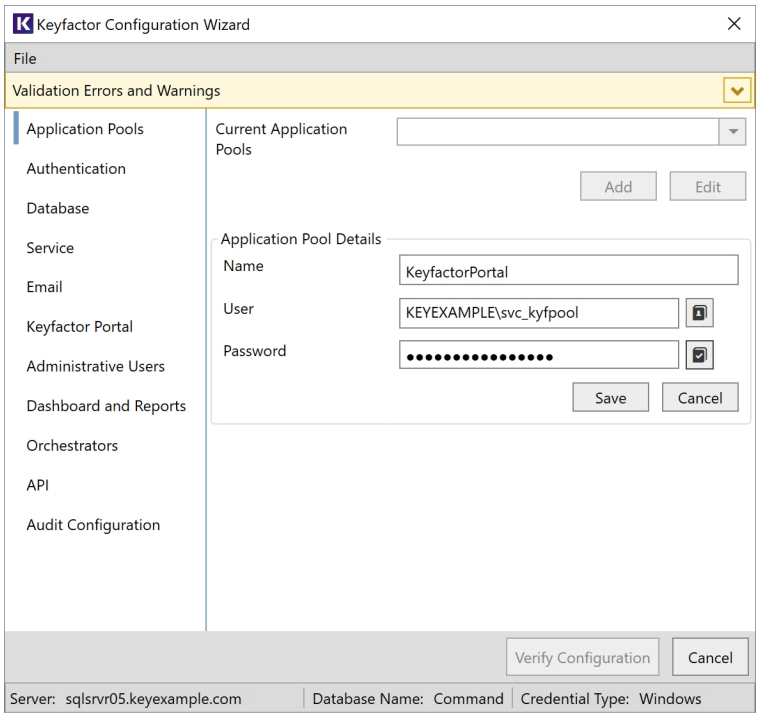


Figure 509: Configure: Application Pools

13. **Authentication Tab**

On the Authentication tab of the configuration wizard, check the **Use OAuth** box if you wish to use an identity provider other than Active Directory as either your primary identity provider or a federation gateway to another identity provider. If you do not select this, you will be using the default identity provider of Microsoft Active Directory. If you checked OAuth, configure it as follows.

Important: Only one identity provider may be configured at a time on each Keyfactor Command instance.

On the Authentication tab in the top section, accept the defaults for the **Session Expiration** and **Cookie Expiration** or modify these if appropriate for your environment. The *Session Expiration* value determines the length of time a browser session in the Keyfactor Command Management Portal will remain logged in before the user is prompted to re-authenticate regardless of whether the session is idle or in active use. The *Cookie Expiration* value determines the length of time the authentication cookie for the Keyfactor Command Management Portal browser session is considered valid. After half of the setting's duration, Keyfactor Command will attempt to use a refresh token to update the cookie. If this fails, the user's session will be terminated. The cookie renewal is seamless from the user's perspective (there is no prompt for credentials).

Note: For Keyfactor Identity Provider, these values should match those configured for the *SSO Session Max* and *Access Token Lifespan* in Keyfactor Identity Provider (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). If you've opted not to issue refresh tokens in Keyfactor Identity Provider, the **Cookie Expiration** value should match the **Session Expiration** value.

Claims Proxy Section

On the Authentication tab in the Claims Proxy section, enter the FQDN that you will use to access the Keyfactor CommandManagement Portal in the **Host Name** field. This can be either the actual host name of the server on which you are installing the Keyfactor Command Administration component or a DNS alias pointing to the server. If you have multiple Keyfactor CommandManagement Portal servers with load balancing, this will be a DNS name pointing to your load balancer. Select the Default Web Site in the **Web Site** dropdown, or other web site as desired. Accept the default for the **Virtual Directory** and select the application pool for this virtual directory that you created earlier in the **Application Pool** dropdown.

Note: When you install Keyfactor Command using an identity provider other than Active Directory, a virtual directory called, by default, KeyfactorProxy is created in IIS for OAuth authentication. If you later disable OAuth, this virtual directory will still exist. This virtual directory is not used if you opt to use Active Directory as an identity provider. During Keyfactor Command uninstallation it will be removed, or you may opt to remove it manually if you are switching from OAuth to Active Directory for authentication.

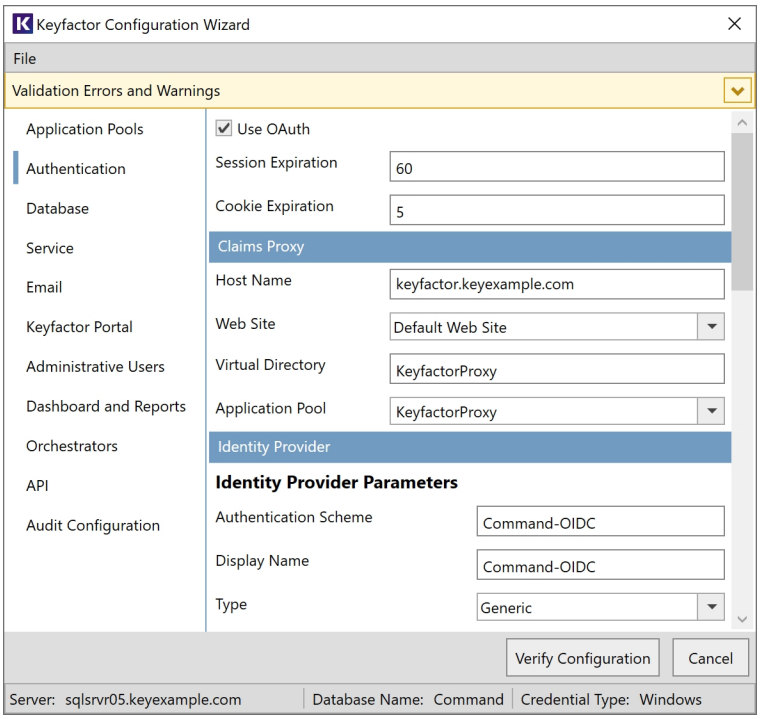


Figure 510: Configure: Identity Providers—OAuth Claims Proxy Section

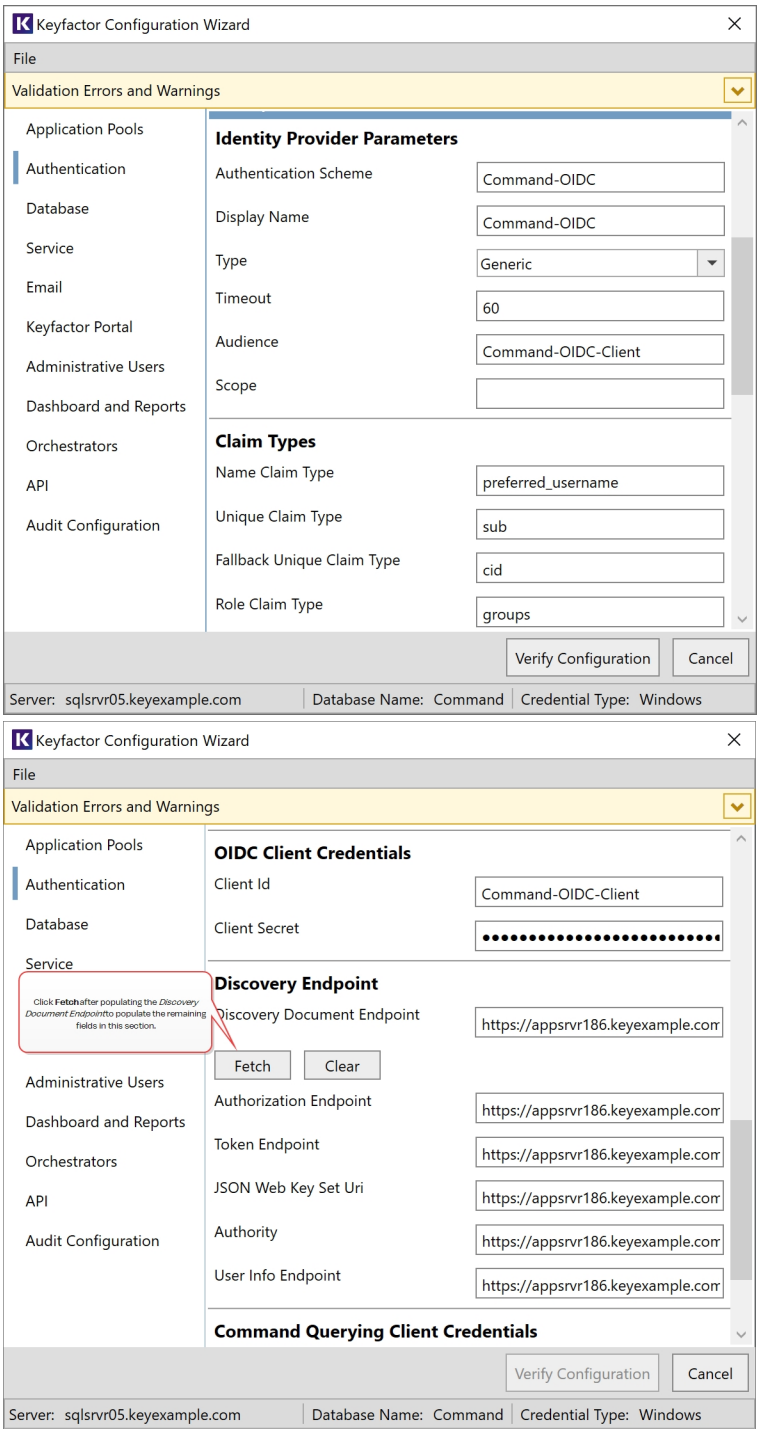
Tip: There is no *Use SSL* check box for this component because SSL is required.

Identity Provider Section

On the Authentication tab in the Identity Provider section, enter an **Authentication Scheme** and **Display Name** for your identity provider. The Authentication Scheme should be entered without spaces. This is used in constructing URLs that reference the identity provider from Keyfactor Command. In the **Type** dropdown, select an appropriate type for your identity provider. Most identity providers can be supported with the *Generic* type. For Auth0, select the *Auth0* type.

For Keyfactor Identity Provider, the Authentication Scheme you enter here must match the name you used when configuring the redirect URIs for Keyfactor Identity Provider (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

Populate the identity provider parameters according to Table 849: Identity Provider Parameters and click **Save**. The fields that appear will vary depending on the selected *Type*. HTTPS is required for URL parameters. For more information about identity providers, see Selecting an Identity Provider for Keyfactor Command on page 2704.


Keyfactor Configuration Wizard
File
Validation Errors and Warnings
Application Pools
Authentication
Database
Service
Email
Keyfactor Portal
Administrative Users
Dashboard and Reports
Orchestrators
API
Audit Configuration
Identity Provider Parameters
Authentication Scheme
Command-OIDC
Display Name
Command-OIDC
Type
Generic
Timeout
60
Audience
Command-OIDC-Client
Scope
Claim Types
Name Claim Type
preferred_username
Unique Claim Type
sub
Fallback Unique Claim Type
cid
Role Claim Type
groups
Verify Configuration
Cancel
Server: sqlsrvr05.keyexample.com
Database Name: Command
Credential Type: Windows
Keyfactor Configuration Wizard
File
Validation Errors and Warnings
Application Pools
Authentication
Database
Service
Click Fetch after populating the Discovery Document Endpoint to populate the remaining fields in this section.
Administrative Users
Dashboard and Reports
Orchestrators
API
Audit Configuration
OIDC Client Credentials
Client Id
Command-OIDC-Client
Client Secret
Discovery Endpoint
Discovery Document Endpoint
https://appsrvr186.keyexample.com
Fetch
Clear
Authorization Endpoint
https://appsrvr186.keyexample.com
Token Endpoint
https://appsrvr186.keyexample.com
JSON Web Key Set Uri
https://appsrvr186.keyexample.com
Authority
https://appsrvr186.keyexample.com
User Info Endpoint
https://appsrvr186.keyexample.com
Command Querying Client Credentials
Verify Configuration
Cancel
Server: sqlsrvr05.keyexample.com
Database Name: Command
Credential Type: Windows

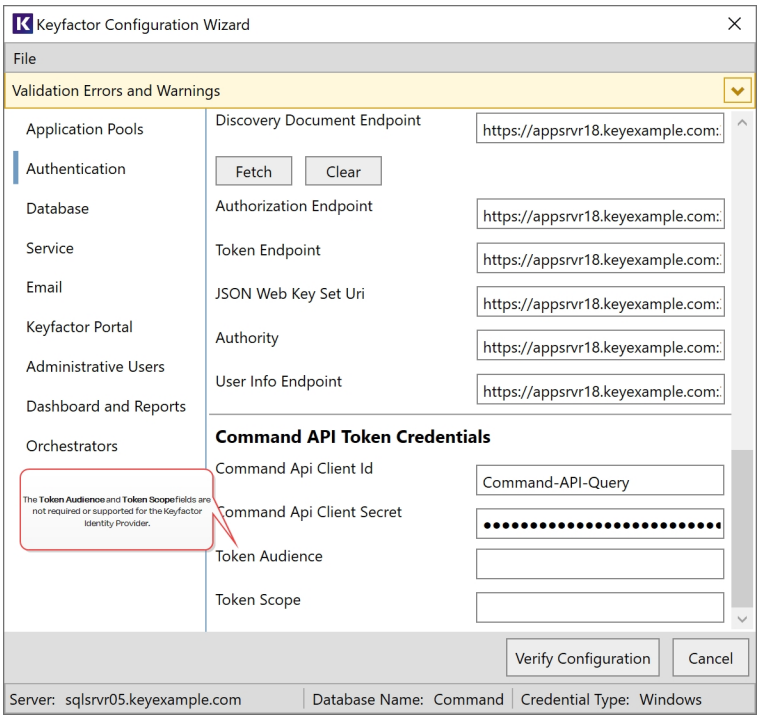


Figure 511: Configure: Identity Providers—OAuth Identity Provider Section

Note: If you return to the configuration wizard and re-run it to add a new identity provider or change the identity provider that's in active use for the system (disable one and enable another), you should restart the web server services (run an iisreset) after making the change to clear any cached data. If you're using more than one Keyfactor Command server in a cluster configuration, the web server services should be restarted on all of them.

Table 849: Identity Provider Parameters

Name	Type	Example	Description
Admin Querying Client Id	1 - String	Command-API-Query	The client ID of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). Keyfactor recommends that you use a different client for this purpose than the client

Name	Type	Example	Description
			used for the main connection from Keyfactor Command to the identity provider (see *Client Id*). This parameter is required.
Admin Querying Client Secret	1 - String		The client secret of the service account that Keyfactor Command uses to make API calls to the identity provider. For Keyfactor Identity Provider, this is created as a client (see Service Accounts on page 2730). This parameter is required.
Audience	1 - String	Command-OIDC-Client	The audience value for the identity provider. For Keyfactor Identity Provider, this should be set to the same value as the *Client Id*. For example: `Command-OIDC-Client` This parameter is required.
Auth0 API URL	1 - String		The unique identifier defined in Auth0 or a similar identity provider for the API. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Authority	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor	The issuer/authority endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint*

Name	Type	Example	Description
			Fetch step. Review it to confirm that it appears correct. This parameter is required. **Tip:** When you add or update an identity provider, the provider's discovery document is validated based on this authority URL. The discovery document is also validated periodically in the background. The following are validated: • That the discovery document is reachable using the Authority value provided and can be parsed into a valid discovery document. • That the Authority URL matches the Issuer returned in the discovery document. • That all the URLs on the discovery document are using HTTPS. • That the JSONWebKeySetUri value is included on the discovery document. • That any endpoint configuration values (Authorization Endpoint, Token Endpoint, UserInfo Endpoint, JSONWebKeySetUri) that have been saved or are being saved match—including case—the values returned in the discovery document. The UserInfo Endpoint is not a required configuration field, but if a value is provided, it

Name	Type	Example	Description
			must match what's in the discovery document. If any of these validation tests fail, any identity provider changes in process will not be saved and an error will be displayed or logged.
Author-ization Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Key-factor/protocol/openid-connect/auth	The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Client Id	1 - String	Command-OIDC-Client	The ID of the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, this should be: `Command-OIDC-Client` For more information, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). This parameter is required.
Client	2 -		The secret for the client application

Name	Type	Example	Description
Secret	Secret		created in the identity provider for primary application use. For Keyfactor Identity Provider, see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716 for help locating this. It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. Supported methods to store secret information are: • Store the secret information in the Keyfactor secrets table. A Keyfactor secret is a user-defined username or password that is encrypted and stored securely in the Keyfactor Command database. • Load the secret information from a PAM provider. See Privileged Access Management (PAM) on page 742 for more information. This parameter is required.
Discovery Document Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/.well-known/openid-configuration	The discovery URL for the identity provider. For Keyfactor Identity Provider, this is the link to the OpenID Endpoint Configuration page, which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). Populate this value and click **Fetch** to populate the remainder of the fields in this section, if desired. If you opt not to populate this field or if the discovery document does not return

Name	Type	Example	Description
			a valid response, the remainder of the fields in this section of the configuration will need to be configured manually. This value is not stored in the database.
Fallback Unique Claim Type	1 - String	cid	A backup value used to reference the type of claim used for users in the identity provider in case the primary referenced name does not contain a value. This parameter is required.
JSON Web Key Set Uri	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/certs	The JWKS (JSON Web Key Set) URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Name Claim Type	1 - String	preferred_username	The name used to reference the type of user claim for the identity provider. For Keyfactor Identity Provider, this should be: `preferred_username` This parameter is required.
Role Claim Type	1 - String	groups	The value used to reference the type of group claim for the identity provider. For Keyfactor Identity Provider, this should be:

Name	Type	Example	Description
			`groups` This parameter is required.
Scope	1 - String		One or more scopes that are requested during the OIDC protocol when Keyfactor Command is the relying party. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Sign Out URL	1 - String	https://my-auth0-instance.us.auth0.com/oidc/logout	The signout URL for the identity provider. This parameter only appears if *Auth0* is selected as the type and is required in that case.
Timeout	1 - String	60	The number of seconds a request to the identity provider is allowed to process before timing out with an error.
Token Audience	1 - String		An audience value to be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. This value is not used for Keyfactor Identity Provider.
Token Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token	The token endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on

Name	Type	Example	Description
			page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct. This parameter is required.
Token Scope	1 - String		One or more scopes that should be included in token requests delivered to the identity provider when making a token request where Keyfactor Command is acting as the OAuth client. Multiple scopes should be separated by spaces. This value is not used for Keyfactor Identity Provider.
Unique Claim Type	1 - String	sub	The value used to reference the type of claim used for users in the identity provider. For Keyfactor Identity Provider, this should be (for subject): `sub` This parameter is required.
User Info Endpoint	1 - String	https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/certs	The user info endpoint URL for the identity provider. For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). It is automatically returned by the *Discovery Document Endpoint* **Fetch** step. Review it to confirm that it appears correct.
User Query	1 -	https://my-keyidp-serv-	The user query endpoint URL for the

Name	Type	Example	Description
Endpoint	String	er.keyexample.com /admin/realms/Keyfactor	identity provider. For Keyfactor Identity Provider, this is the base URL for the Keyfactor Identity Provider REST API and has an expected value of: `https://<host>/admin/realms/<realm_name>` This parameter is required.

14. **Database Tab**

    On the Database tab in the top section, select an Authentication Mode for ongoing communications to SQL server—**Windows Authentication** or **SQL Server Authentications**. Your SQL server must be configured to support mixed mode authentication in order to use the SQL server authentication option.

    - If you select Windows Authentication, login(s) will be created in SQL for the application pool user(s) you created on the Application Pools tab and granted appropriate permissions (other than the application pool for the Logi Analytics Platform, which does not need database access).
    - If you choose SQL server authentication, enter a **User** and **Password** of a SQL administrator for the Keyfactor Command SQL database. If the user does not exist in SQL, it will be created and granted the necessary permissions for management of the Keyfactor Command database. If the user already exists in SQL, it will be granted the necessary permissions. If the database you originally connected to is an Azure database, **SQL Server Authentication** is the only option provided.

    For more information, see Grant Permissions in SQL on page 2744.

    If desired, check the **Configure Encryption** box. This option allows you to encrypt select sensitive data stored in the Keyfactor Command database using a separate encryption methodology utilizing a Keyfactor Command-defined certificate on top of the SQL server encryption noted above. This additional layer of encryption protects the data in cases where the SQL Server master keys cannot be adequately protected. Read and understand the encryption warning. This warning applies to implementations with more than one Keyfactor Command server.

    Note: In an environment where there are multiple copies of Keyfactor Command pointing to the same database, each server running a Keyfactor Command instance will need to have the same encryption certificate AND the corresponding private key.

Select **Application and SQL** for the **Encryption Type** and click the **Select** button to choose a certificate from the Personal Certificate store of the Local Computer with which to encrypt the data. Only valid certificates with the appropriate key usage will appear in the selection dialog. See Acquire a Public Key Certificate for the Keyfactor Command Server on page 2767.

If you enable application-level encryption, your certificate must either be using a key storage provider (KSP) or you must manually grant permissions to the certificate's private key (see Application-Level Encryption on page 2752).

Tip: If you need to reset the encryption level to remove application-level encryption, run the configuration wizard again and select the **SQL Only** option. You must ensure that the server you are re-running the configuration wizard on has both the certificate used for application-level encryption and its associated private key. When Keyfactor Command notices that application-level encryption has been disabled, it will process all the secrets in the database and remove the additional encryption. The data will then be re-saved to the secrets table using only SQL-level encryption.

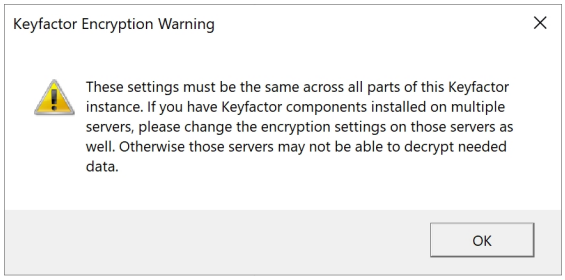


Figure 512: Configure: Encryption Warning

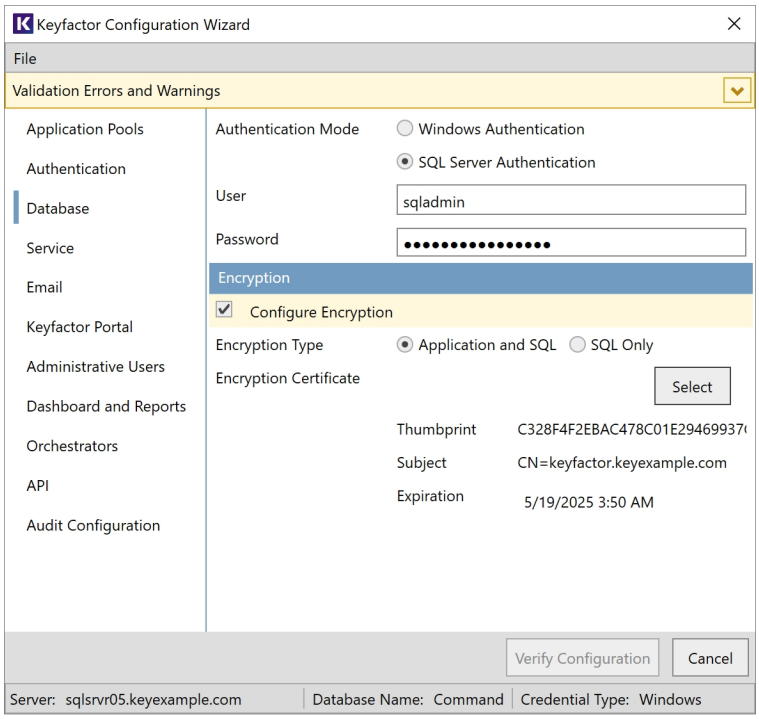


Figure 513: Configure: Database

15. **Service Tab**

    On the Service tab, enter the user name and password of the Active Directory (DOMAIN\username format) or local (HOSTNAME\username format) service account under which the Keyfactor Command Service will run. This can be the same service account used for the application pool(s) or a different service account. You may use the people picker button ( ) to browse for the account. Click the verify button ( ) to confirm that the username and password entered are valid. If desired, check the **Start service on bootup** box to start the Keyfactor Command Service at system start.

    Tip: Checking **Start service on bootup** sets all jobs of type TimerService to *true* in the Keyfactor Command service appsettings.json file. These settings can be modified on a job-by-job basis in this appsettings.json file (see Keyfactor Command Service Job Settings on page 778 for more information).

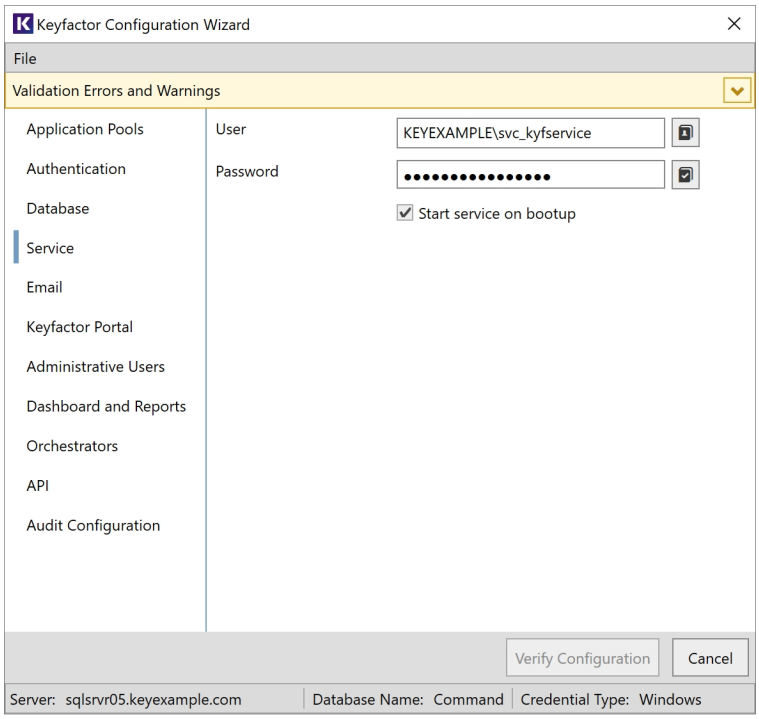


Figure 514: Configure: Service

16. **Email Tab**

    On the Email tab, enter the FQDN of your SMTP server, the SMTP port (default is 25), and the sender name and account. Depending on the email configuration in your environment, the sender account may need to be a valid user on your mail server (using Active Directory credentials) or you may be able to put anything in this field (if your mail server supports anonymous connections). You may use the people picker button ( ) to browse for the sender account if you are using a valid account. Select the **Use SSL** box if this option is supported by your mail server and select the appropriate authentication method for your environment. If your mail server requires that you provide a username and password for a valid user, enter that Active Directory username and password in the fields at the bottom of the page after selecting the **Explicit credentials** radio button. You may use the people picker button ( ) to browse for the account. Click the verify button ( ) to confirm that the username and password entered are valid. The user you select here must match the email address you set in the *Sender Account* field if you select *Explicit credentials*. The information entered on this tab may later be changed in the Keyfactor Command Management Portal.

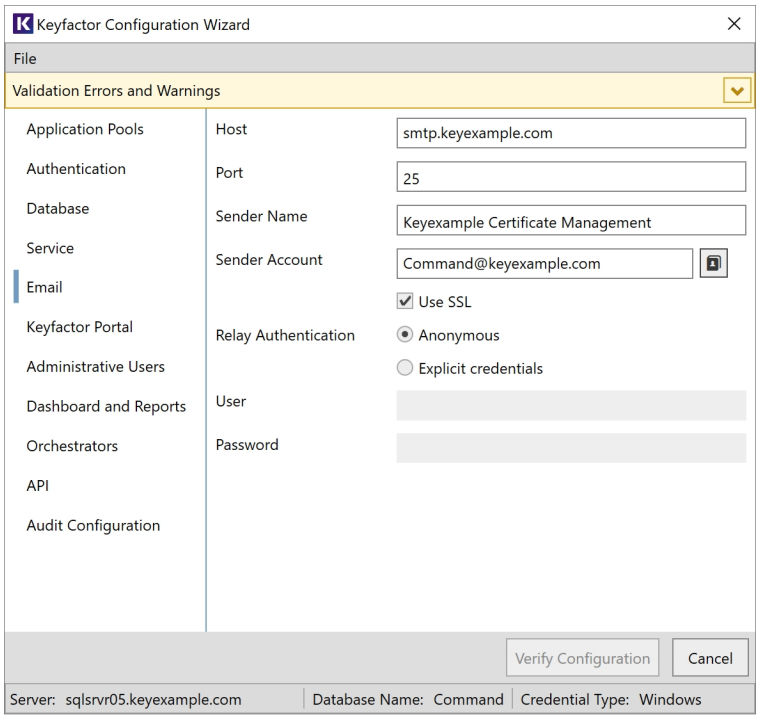


Figure 515: Configure: Email

17. **Keyfactor Portal Tab**

    On the Keyfactor Portal tab in the top section, enter the FQDN that you will use to access the Keyfactor Command Management Portal in the **Host Name** field. This can be either the actual host name of the server on which you are installing the Keyfactor Command Administration component or a DNS alias pointing to the server. If you have multiple Keyfactor Command Management Portal servers with load balancing, this will be a DNS name pointing to your load balancer. Select the Default Web Site in the **Web Site** dropdown, or other web site as desired. Accept the default for the **Virtual Directory** and select the application pool for this virtual directory that you created earlier in the **Application Pool** dropdown. Check or uncheck the **Use SSL** box as appropriate for your environment.

Enrollment Section

In the Enrollment section of the page, modify the default **Certificate Subject Format** field, if desired. The subject values provided in this field are substituted at processing time for any entered by the user in PFX enrollment or provided with enrollment defaults if the template used is set to supply in request.

The data in the subject format takes precedence over any data entered during PFX enrollment or supplied by enrollment defaults (see Enrollment Defaults Tab on page 398). For example, if you define the following subject format:

```
CN={CN},E={E},O=Key Example\, Inc.,OU={OU},L=Chicago,ST=IL,C=US
```

The organization for certificates generated through PFX enrollment will always be *Key Example, Inc.* regardless of what is shown on the PFX enrollment page during enrollment.

This setting also applies to CSRs generated using the CSR generation method.

Data from the default subject *does not* display in the PFX enrollment form. To define defaults that will display in the PFX enrollment form (and can be modified by users), use enrollment defaults (see Enrollment Defaults Tab on page 398).

Note: Backslashes are required before any commas embedded within values in the subject field (e.g. O=Key Example\, Inc.). Quotation marks should not be used in the strings in the fields except in the case where these are part of the desired subject value, as they are processed as literal values.

Tip: The default subject format *does not* apply to enrollments done using the CSR enrollment method or any requests done with the Keyfactor API.

PFX Enrollment Section

In the PFX Enrollment section of the page, uncheck the **Enabled** box if you do not wish to support PFX enrollment. If you wish to support PFX enrollment, leave the **Enabled** box checked. Select the **Domain** radio button if you wish PFX files to be protected with the user's Active Directory password or select the **Auto-Generated** radio button if you wish PFX files to be protected with a one-time password. Check the **Alphanumeric Password Characters** box if you wish the one-time password used to protect PFX files to contain numbers and letters. Uncheck the **Alphanumeric Password Characters** box if you wish the one-time password used to protect PFX files to contain numbers, letters and special characters. In the **Password Length** field, enter a number for the number of characters the one-time password should have. The minimum value is 8. If you select the **Domain** radio button, the data entered in the password fields is not relevant.

CSR Enrollment Section

In the CSR Enrollment section of the page, uncheck the **Enabled** box if you do not wish to support CSR enrollment. If you wish to support CSR enrollment, leave the **Enabled** box checked.

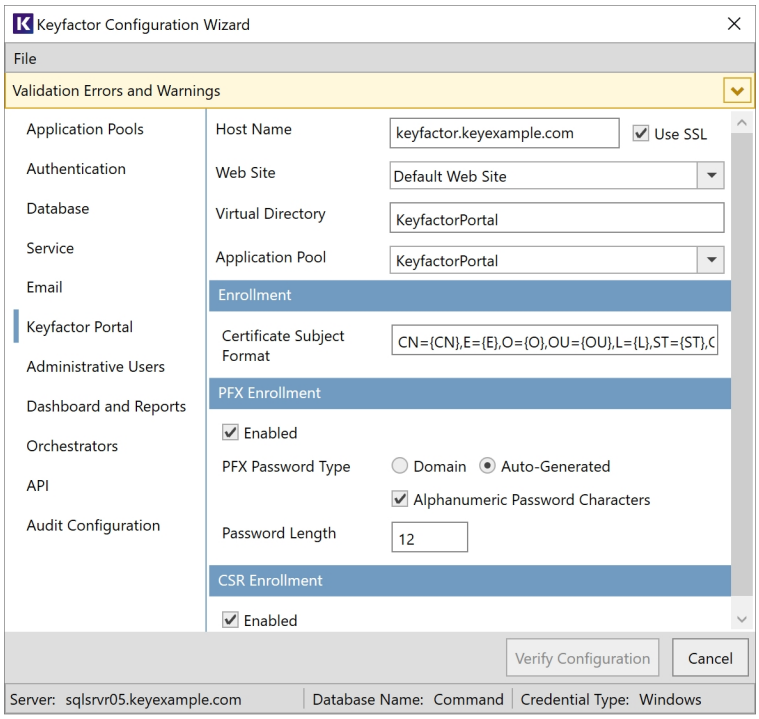


Figure 516: Configure: Keyfactor Portal

18. **Administrative Users Tab**

    On the Administrative Users tab, click **Add** to add users or groups that you will use to control administrative access to the Keyfactor Command Management Portal.

    Enter only the users and/or group(s) to which you want to grant full administrative rights to the Keyfactor Command Management Portal. Following initial configuration, you can create other permission levels and grant those permission levels to other users or groups through the Keyfactor Command Management Portal. See Security Roles and Claims on page 622 for more information.

Users and Groups for Active Directory as an Identity Provider

For each user to be added:

- In the **Identity Provider** dropdown, select Active Directory.
- In the **Claim Type** dropdown, select ADUser for a user account or ADGroup for a group created in Active Directory.
- In the **Claim Value** field enter the user or group name for the account in DOMAIN\name format (e.g. KEYEXAMPLE\Keyfactor Administrators).
- In the **Description** field enter a description to help you identify the user or group (e.g. the user's name).

Important: The built-in Active Directory groups Domain Admins and Enterprise Admins cannot be used directly to grant access to the Management Portal due to how these groups function within Windows. You can create a custom Active Directory group, reference that group in the Management Portal, and add the built-in Domain Admins or Enterprise Admins group to that custom group, if desired.

Important: For environments using Active Directory as an identity provider, the administrative group must be created as a Global or Universal group. If the administrative group is created as a domain-local group, it will not be recognized by the Management Portal Security Roles and Identities configuration. Configuration of the Management Portal will be incomplete until the group is deleted and recreated as a Global or Universal group.

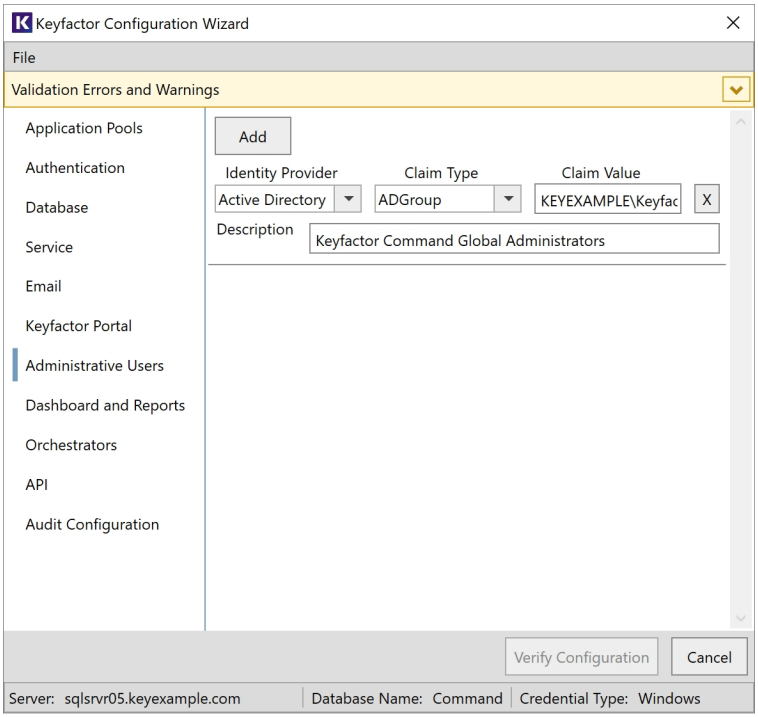


Figure 517: Configure: Administrative Users for Active Directory

Users for an Identity Provider Other Than Active Directory

For each user to be added:

- In the **Identity Provider** dropdown, select OAuth.
- In the **Claim Type** dropdown, select OAuthSubject for a user account created in your OAuth identity provider, OAuthRole for a role created in your OAuth identity provider, or OAuthClientId for a client application created in your OAuth identity provider. If your

claim doesn't fall into one of these categories, select OAuthOid.

- In the **Claim Value** field enter the GUID for the user account, role name for the role, or client ID for the client (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). If you selected OAuthOid, enter an appropriate ID to identify the claim.
- In the **Description** field enter a description to help you identify the user (e.g. the user or group name).

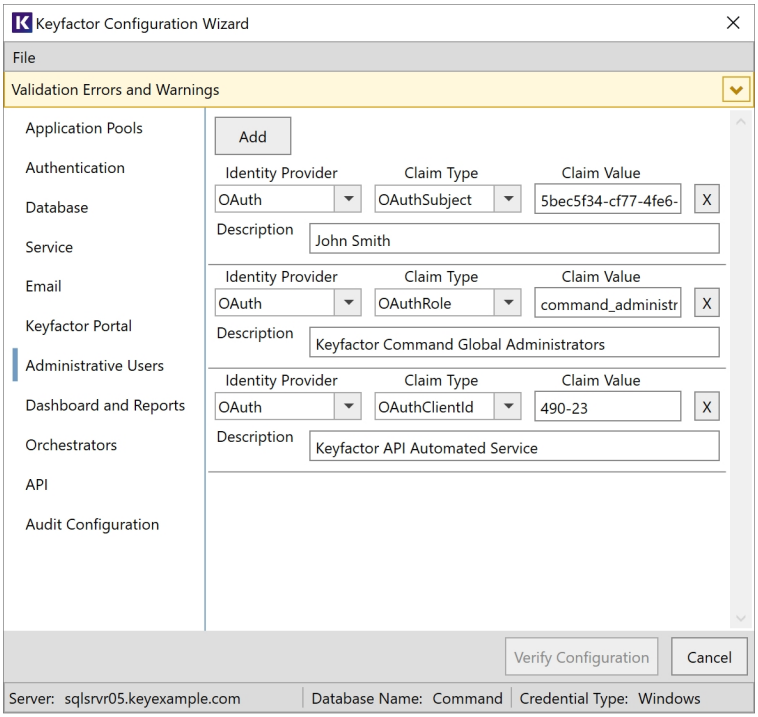


Figure 518: Configure: Administrative Users for OAuth

19. **Dashboard and Reports Tab**

    On the Dashboard and Reports tab, enter the FQDN of the server hosting the Keyfactor Command Management Portal—where the Logi Analytics Platform is installed—in the **Host Name** field. This can be either the actual host name of the server on which you are installing the Keyfactor Command Management Portal component or a DNS alias pointing to the server. Check or uncheck the **Use SSL** box as appropriate for your environment. Select the Default Web Site in the **Web Site** dropdown, or other web site as desired. Accept the default for the **Virtual Directory** and select the application pool for this virtual directory that you created earlier in the **Application Pool** dropdown.

Note: If you are installing the Management Portal in a load balanced configuration, see Appendix - Logi Load Balancing: Keyfactor Command Configuration Wizard Setup on page 2872.

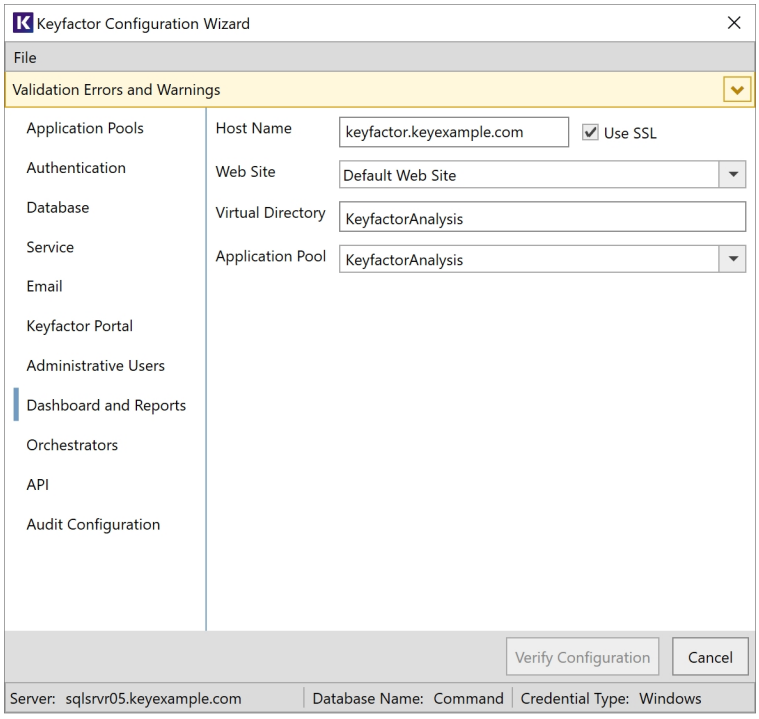


Figure 519: Configure: Dashboard and Reports

20. Orchestrators

    On the Orchestrators tab, enter the FQDN of the server hosting the Keyfactor Command orchestrators web site in the **Host Name** field. This can be either the actual host name of the server on which you are installing the Keyfactor Command Services (Orchestrator Services API) components or a DNS alias pointing to the server. Select the Default Web Site in the **Web Site** dropdown, or other web site as desired. Accept the default for the **Virtual Directory** and select the application pool for this virtual directory that you created earlier in the **Application Pool** dropdown. Check or uncheck the **Use SSL** box as appropriate for your environment.

Reenrollment Section (Optional)

In the **Template For Submitted CSRs** field, from the dropdown, select the template to be used for reenrollment requests made from the Certificate Stores page.

In the **CA For Submitted CSRs** field, enter the certificate authority used for reenrollment requests made from the Certificate Stores page. The CA should be entered in the format FQDN\Logical Name.

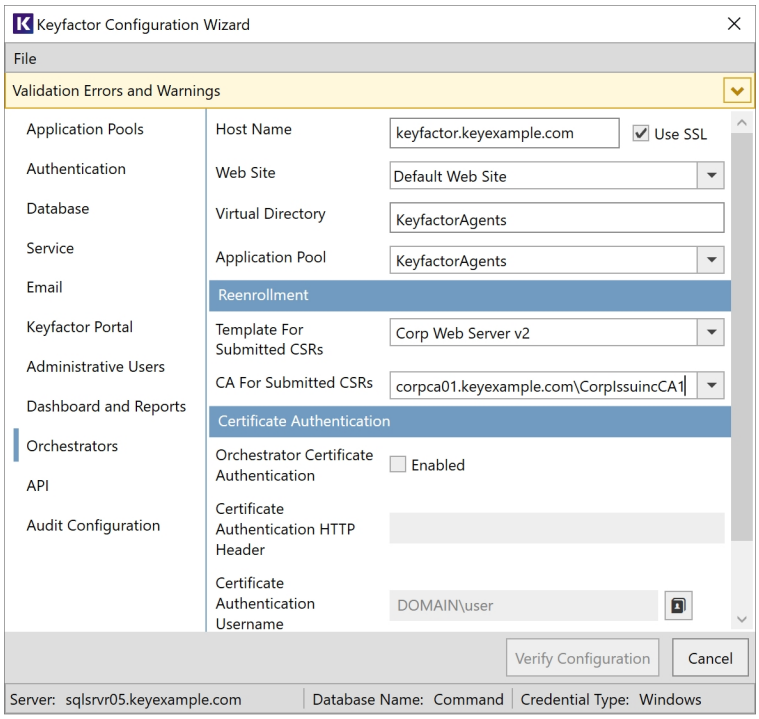


Figure 520: Configure: Orchestrators with Standard Authentication

Certificate Authentication Section (Optional)

In the Certificate Authentication section of the Orchestrators tab, check the **Enabled** box if you wish to support client certificate enrollment from the Keyfactor Universal Orchestrator. In the **Certificate Authentication HTTP Header** field, enter the HTTP header under which the orchestrator connection proxy should send the client authentication certificate. Keyfactor Command uses the certificate supplied in this header to identify the orchestrator attempting to authenticate. In the **Certificate Authentication Username** and **Certificate Authentication Password** fields, enter the credentials for the Active Directory user configured on the proxy to authenticate the orchestrator(s) to the Keyfactor Command server.

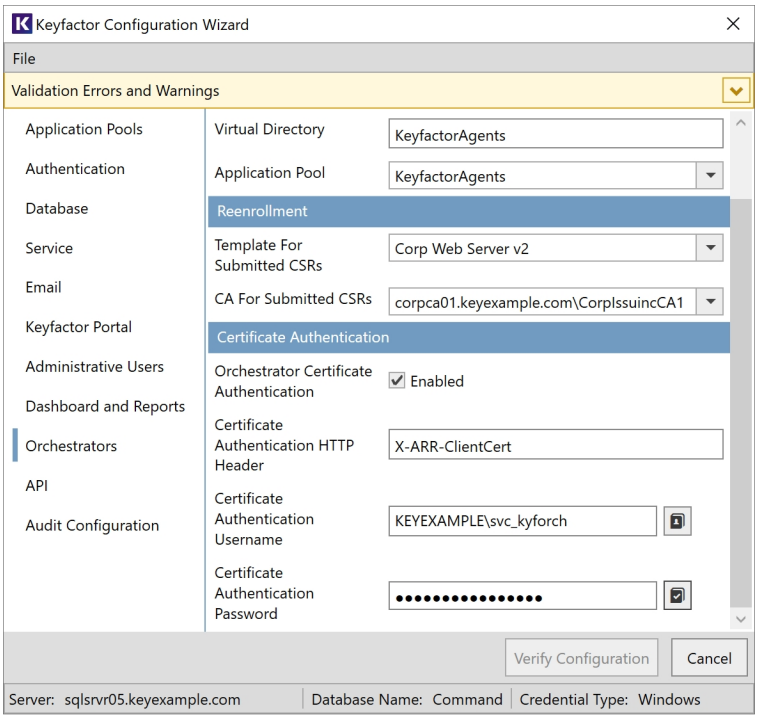


Figure 521: Configure: Orchestrators with Client Certificate Authentication

21. **API Tab**

On the API tab, enter the FQDN of the server hosting the Keyfactor Command KeyfactorAPI service in the **Host Name** field. This can be either the actual host name of the server on which you are installing the Keyfactor Command Services (Keyfactor API) components or a DNS alias pointing to the server. Select the Default Web Site in the **Web Site** dropdown, or other web site as desired. Accept the default for the **Virtual Directory** and select the application pool for this virtual directory that you created earlier in the **Application Pool** dropdown. Check or uncheck the **Use SSL** box as appropriate for your environment.

Tip: Keyfactor Command includes embedded documentation which includes links to the Keyfactor API Reference and Utility from within the documents. If the API is installed in a non-standard virtual directory, these links will not work from the documentation. Keyfactor, recommends accepting the default virtual directory during installation of Keyfactor Command to avoid issues with this.

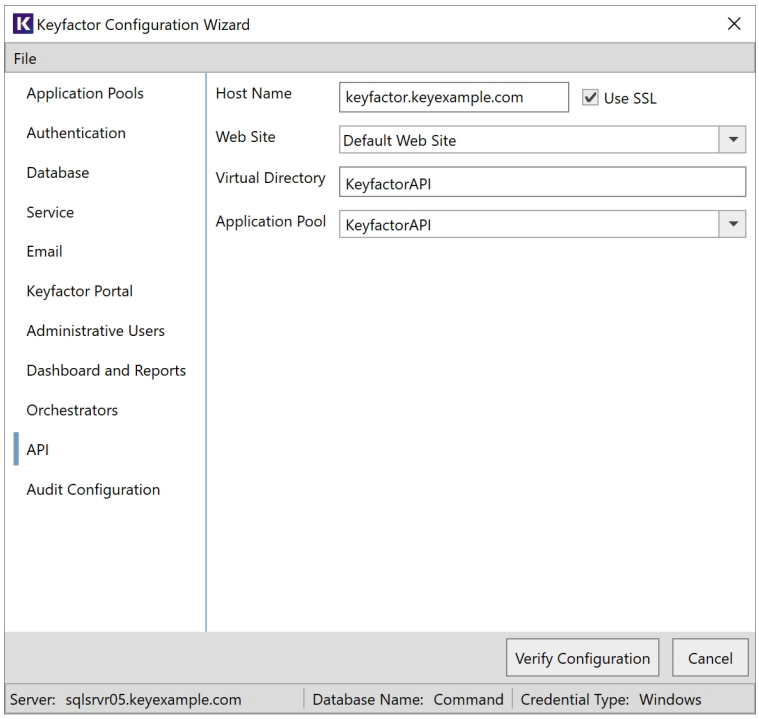


Figure 522: Configure: API

22. **Auditing Configuration Tab**

    On the Auditing Configuration tab, enter the number of years to retain audit data in the **Audit Entry Retention Period (years)** field. By default, seven years of data is retained. The audit log cleanup job runs once daily and removes any audit log entries older than the time specified in the retention parameter except those in the following protected categories:

    - Security
    - CertificateCollections
    - ApplicationSettings
    - SecurityIdentities
    - SecurityRoles

Linux SysLog Server Section

In the Linux SysLog Server section of the page, check the **Connect to SysLog** to enable the option to copy audit logs in real time to a separate server for collection and analysis with a centralized logging solution (e.g. rsyslog, Splunk, Elastic Stack). In the **Host Name** field, enter the fully qualified domain name of the server that will be receiving the logs. Set the **Port** to the port on which your log receipt application is listening to receive the logs. The default value is 514 (the default rsyslog port). If desired, turn on **Use TLS SysLogging**. When you click **Save**, Keyfactor Command will verify that a connection can be made to the specified server on the

specified port. Additional configuration on both the Keyfactor Command server and log receipt server are needed to make TLS communications work (see Prepare for External Log Shipping over TLS (Optional) on page 2775). If you have not yet completed these configurations, you will receive a validation error on save if the Use TLS SysLogging option is enabled.

The auditing settings can be updated on the auditing tab of the applications settings page following installation (see Application Settings: Auditing Tab on page 608).

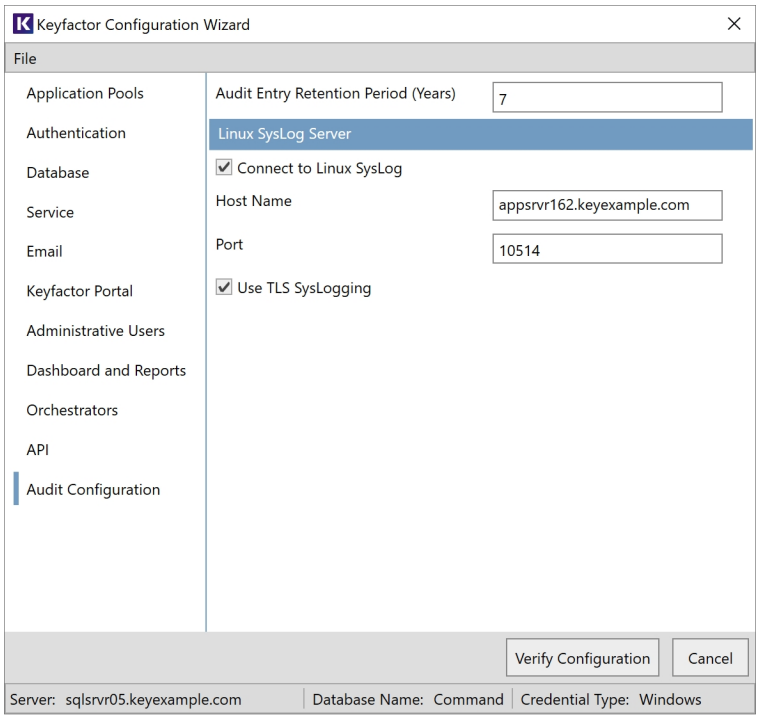


Figure 523: Configure: Audit

23. At this point in the configuration, if you have populated all the required fields, the yellow warning banner at the top of the configuration wizard should have disappeared. If it is still visible, click the dropdown arrow to open the Warnings page and review the warning(s) to see what needs to be corrected. Under some circumstances you will be allowed to continue with the configuration even if the yellow warning banner is still present. You will know this is the case if the **Verify Configuration** button is active. Under these circumstances, you should review the warnings before continuing.

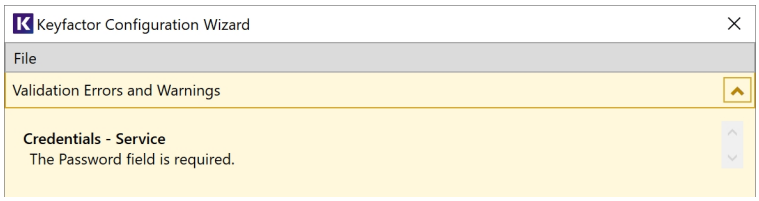


Figure 524: Configure: Configuration Warnings

24. Before completing the configuration wizard, you may choose to save a copy of the configuration as a file for future use. To download the configuration as a file, in the configuration wizard, click **File** at the top of the wizard and choose **Save Data File**. Browse to a location where you want to save the configuration file, enter a file name and click **Save**. You will be prompted to enter a password to encrypt the data in the file. You may choose to protect the file with a password or not. If you use a password at this time, you will need to provide this password to open the file. Keyfactor highly recommends using a strong password to protect the file. If you do not wish to use a password to protect the file, sensitive information (e.g. passwords for the service accounts entered in the configuration wizard) will be removed from the file. Once you enter a password or uncheck the encryption box, click **OK** to save the file.

    Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

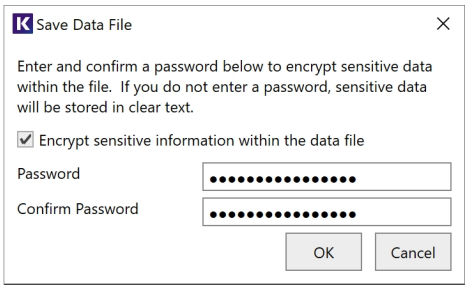
    

    Figure 525: Configure: Save Configuration as a File

25. At the bottom of the Keyfactor Command Configuration Wizard dialog, click **Verify Configuration**.

26. On the Configuration Operations page, review the planned operations and then click **Apply Configuration**. Prior to clicking **Apply Configuration**, you can revisit any of the Configuration Wizard tabs to review or make changes by clicking **Edit Configuration**.

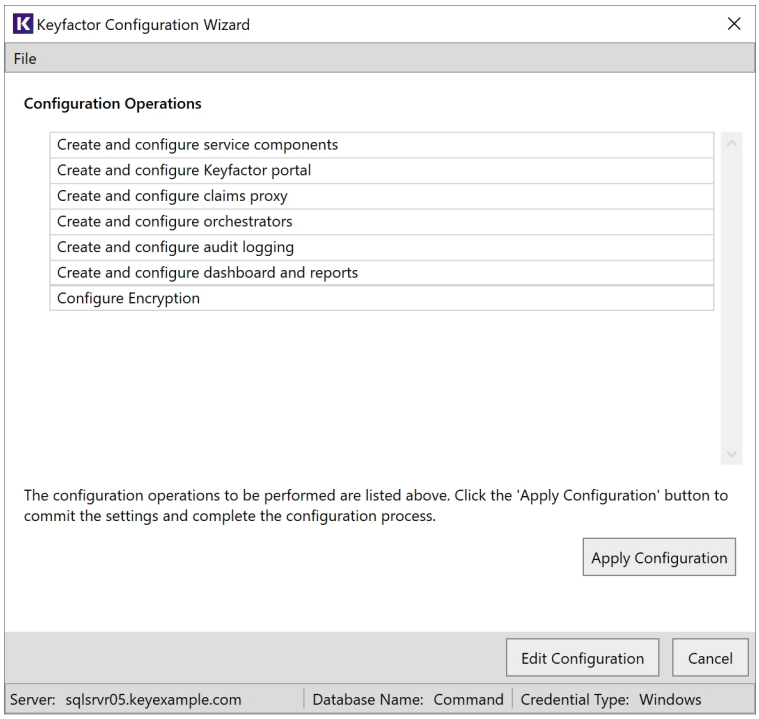


Figure 526: Configure: Configuration Operations

27. When the configuration completes successfully, you will see the below message. If you didn't save a copy of the configuration earlier, you may do so at this time by clicking **Save Settings**. Otherwise, click **Close** to close the dialog.

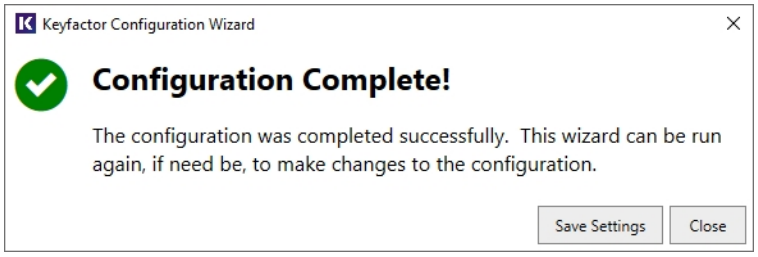


Figure 527: Configure: Configuration Complete

4.4.3.2 Install the Keyfactor Command Server from the Command Line

The Keyfactor Command server can optionally be configured using a pair of configuration files and a command run from the command line. You may be provided one or both of these files by your Keyfactor Customer Success Manager. The configuration files for command-line configuration are:

- Keyfactor Command Configuration File

  This file, with an extension of .cmscfg, contains information in XML format to configure the Keyfactor Command database. This file can be generated by installing Keyfactor Command,

running the configuration wizard and populating all the fields as desired, and then saving a copy of the configuration either with or without a password to encrypt sensitive information in the file (see Install the Keyfactor Command Components on the Keyfactor Command Server(s) on page 2780). Keyfactor highly recommends using a strong password to protect the file. A file that has not been protected with a password will be missing the sensitive information that would be protected by the password encryption (e.g. service account passwords).

- Input Parameters File

  This file, with an extension of .xml, contains information in XML format to connect to and configure SQL, open the Keyfactor Command configuration file, locate the Keyfactor Command license, and create application pools, if desired.

To configure and, optionally, install Keyfactor Command from the command line:

1. Install the Keyfactor Command software using one of these methods:

   - Follow the initial instructions for Install the Keyfactor Command Components on the Keyfactor Command Server(s) on page 2780 except on the final installation wizard page, uncheck the *Launch the Configuration Wizard now* box and click **Finish**. The configuration wizard should not open.
   - Open an administrative command prompt and execute a command similar to the following:

     ```
     start /wait msiexec /i <full path to install file>\KeyfactorPlatform.msi /Live
     <path for msiexec logs> /Quiet
     ```

     This will install the default components of Keyfactor Command in a non-interactive way (/Quiet), output log information to a file (/Live), and wait to return to the command prompt until the installation is complete (start /wait).

     If you wish to install a set of features other than the default features, you can add the ADDLOCAL parameter and specify the features you wish to install. For example, the following command will install the *Orchestrator Service API* and *Windows Services* features:

     ```
     start /wait msiexec /i <full path to install file>\KeyfactorPlatform.msi
     ADDLOCAL=AgentServicesFeature,ServiceFeature /Live <path for msiexec logs>
     /Quiet
     ```

     The following features are available:

     - AgentServicesFeature

       This installs the Orchestrator Service API feature.
     - ConfigurationFeature

       This installs the configuration wizard and is required for all installations.
     - ServiceFeature

       This installs the Windows Services feature, which includes the Keyfactor Command Service (a.k.a. the timer service).
     - VCRedistFeature

This installs the Microsoft Visual C++ Redistributable and is required for all installations unless it has been separately installed.

- WebApiFeature

  This installs the WebAPI feature, which includes the Keyfactor API.

- WebConsoleFeature

  This installs the Management Portal feature, which includes the Keyfactor Command Management Portal and the Keyfactor API.

The features you decide to install will depend on the role the server will be playing in your Keyfactor Command implementation. Table 850: Features Required for Each Server Role shows the minimum features that need to be installed for each of the server roles shown in the table columns. If you're installing all the required features on a single server, you need everything. If you don't intend to use any orchestrators (see Installing Orchestrators on page 2875), you do not need to install the *AgentServicesFeature*.

Table 850: Features Required for Each Server Role

ADDLOCAL Parameter	Single Server	Management Portal	Windows Services	Keyfactor API	Orchestrator Service API
ConfigurationFeature	✓	✓	✓	✓	✓
VCRedistFeature	✓	✓	✓	✓	✓
WebConsoleFeature	✓	✓			
ServiceFeature	✓		✓		
WebApiFeature				✓	
AgentServicesFeature	✓				✓

2. Acquire a Keyfactor Command configuration file from your Keyfactor Customer Success Manager or create one by installing and configuring Keyfactor Command on a test machine. It's not practical to attempt to generate this file manually, though a file can be edited once generated (other than password-protected fields).

3. Create an input parameters file. See Table 851: Input Parameters XML File Fields. A sample file can be found in the Configuration directory under the directory in which you installed Keyfactor Command. By default, this is:

   C:\Program Files\Keyfactor\Keyfactor Platform\Configuration\InputParameters.xml

4. Open an administrative command prompt, change to the Configuration directory under the directory in which you installed Keyfactor Command (by default this is C:\Program Files\Keyfactor\Keyfactor Platform\Configuration), and execute a command similar to the following, referencing your input parameters file and using the appropriate parameters for the ConfigurationWizardConsole tool (see Table 852: ConfigurationWizardConsole.exe Options):

    ```
    .\ConfigurationWizardConsole.exe -p C:\Stuff\InputParameters.xml -u
    ```

Tip: Check the Keyfactor Command log and the Windows application event log for errors if the installation does not complete successfully (see Configure Logging on page 2829).

Table 851: Input Parameters XML File Fields

Parameter	Description
Protected	A Boolean indicating whether sensitive information in the Keyfactor Command configuration file is protected with a password (true) or not (false).
Password	A string containing the password used to protect the Keyfactor Command configuration file if *Protected* is set to **true**.
ConfigurationFile	The full path to the Keyfactor Command configuration file (e.g. C:\Stuff\-myconfig.cmscfg).
DatabaseServer	The hostname or IP address of the SQL server where the Keyfactor Command database will be installed, with optional port. For example: • Local with default port: mysql.keyexample.com • Azure SQL myazuresql.database.windows.net,1433
Database	The name of the database in SQL for Keyfactor Command. If a database with this name exists, it will be used (see *ForceDatabaseConversion*). If it doesn't, it will be created (see *CreateDatabaseIfMissing*).
CreateDatabaseIfMissing	A Boolean indicating whether the SQL database should be created if it does not exist (true) or not (false). If this is set to **false** and a database does not exist, an error will be generated and the configuration will not continue.
ForceDatabaseConversion	A Boolean indicating whether a pre-existing SQL database should be converted for use by Keyfactor Command (true) or not (false). If this is set to **false** and a pre-existing database that has not already been converted for Keyfactor Command use is found, an error will be generated and the configuration will not continue.
ForceDatabaseUpgrade	A Boolean indicating whether a pre-existing SQL database should be upgraded from a previous version of Keyfactor Command (true) or not (false). If this is set to **false** and a pre-existing database that is running a version of Keyfactor Command that does not match the version being installed is found, an error will be generated and the configuration will not continue.
ContinueOnSqlGrantError	A Boolean indicating whether the configuration should continue if an error is encountered when attempting to set SQL permissions.
SqlUsername	A string containing the SQL username to be used to authenticate to the SQL server if you have opted to use SQL authentication. For an on-

Parameter	Description
	premise SQL server, the server must be configured to support mixed mode authentication in order to use the SQL option. This option can be used to connect to cloud-based (e.g. Azure) SQL servers. Leave this field blank if you are using Windows integrated authentication. The credentials of the logged on user executing the command will be used to authenticate to SQL.
SqlPassword	A string containing the SQL password to be used to authenticate to the SQL server. Leave this field blank if you are using Windows integrated authentication.
LicenseFile	The full path to your Keyfactor Command license file (e.g. C:\Stuff\keyexample.cmslicense).
AppliationPoolsToCreate	An array of application pools to create. A separate application pool is required for each virtual directory that will be created for Keyfactor Command in IIS. If you choose to install all the roles, this will be either four or five application pools for the virtual directories with the following names, by default: • KeyfactorAgents (Keyfactor Command agent and orchestrator service endpoint) • KeyfactorAnalysis (Keyfactor Command dashboard and reporting) • KeyfactorAPI (Keyfactor API) • KeyfactorPortal (Keyfactor Command Management Portal) • KeyfactorProxy (Keyfactor Command proxy to your identity provider for OAuth support; only created if an identity provider other than Active Directory is used as the identity provider) Application pool fields include:

Parameter	Description
Name	A string containing the name of the application pool to create.
Username	A string containing the user name of the Active Directory (DOMAIN\username format) or local (HOSTNAME\username format) service account under which the application pool will run.  **Tip:** The same service account may be used for all application pools.

Parameter	Description

	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>Password</td><td>A string containing the password of the Active Directory or local service account under which the application pool will run.</td></tr><tr><td>FailIfExists</td><td>A Boolean indicating whether the configuration will fail if the application pool already exists (true) or not (false).</td></tr></table>
	For example:
	<pre><ApplicationPoolsToCreate>
	<!--Remove this section if none are to be created-->
	  <WizardApplicationPool>
	  <Name>KeyfactorPortalPool</Name>
	  <Username>KEYEXAMPLE\svc_kyfpools</Username>
	  <Password>MySecurePassword</Password>
	  <FailIfExists>true</FailIfExists>
	  </WizardApplicationPool>
	  <WizardApplicationPool>
	  <Name>KeyfactorAPIPool</Name>
	  <Username>KEYEXAMPLE\svc_kyfpools</Username>
	  <Password>MySecurePassword</Password>
	  <FailIfExists>true</FailIfExists>
	  </WizardApplicationPool>
	  <WizardApplicationPool>
	  <Name>KeyfactorAnalysisPool</Name>
	  <Username>KEYEXAMPLE\svc_kyfpools</Username>
	  <Password>MySecurePassword</Password>
	  <FailIfExists>true</FailIfExists>
	  </WizardApplicationPool>
	  <WizardApplicationPool>
	  <Name>KeyfactorAgentsPool</Name>
	  <Username>KEYEXAMPLE\svc_kyfpools</Username>
	  <Password>MySecurePassword</Password>
	  <FailIfExists>true</FailIfExists>
	  </WizardApplicationPool>
	</ApplicationPoolsToCreate></pre>

Table 852: ConfigurationWizardConsole.exe Options

Switch	Description
-p, --paramfile	The full path to the input parameters XML file. This switch is **required**.
-u, --unattended	Do not output errors at the console. Errors will be redirected to the Windows event log.
-d, --database	Create the database in SQL but do not configure Keyfactor Command.
-s, --scriptpath	The full path to a non-standard location for the scripts used during a database upgrade. By default, these are found in the following path: `C:\Program Files\Keyfactor\Keyfactor Platform\Configuration\DatabaseUpgrade` This option is typically only used by Keyfactor Support.
--help	Display the help.
--version	Display the version information.

4.4.4 Initial Configuration

Once the installation and configuration wizards are complete, only a few configuration tasks remain before Keyfactor Command will be up and running at a basic level. This section details the basic post-install configuration steps that need to be completed to get Keyfactor Command up and running. See the *Keyfactor Command Reference Guide* for more advanced configuration guidance. See the separate installation guides for client components such as the Keyfactor Universal Orchestrator, Keyfactor Java Agent, and Keyfactor CA Gateways.

After you have completed all the steps in this guide, the certificate search and report functions in the Keyfactor Command Management Portal should be functioning. Further configuration, as described in the *Keyfactor Command Reference Guide*, is required to make these features function:

- Using the Keyfactor Command Management Portal Dashboard on page 6
- Configuring Enrollment through the Keyfactor Command Management Portal (see Certificate Template Operations on page 381)
- Security Roles and Claims on page 622 for the Keyfactor Command Management Portal
- Revocation Monitoring on page 210, Expiration Alerts on page 167 and Pending Certificate Request Alerts on page 178
- Using the Workflow Builder (see Workflow on page 229)
- External Certificate Synchronization with SSL Discovery on page 453 and Certificate Stores on page 408
- Managing SSH on page 525 Keys

4.4.4.1 Configure Kerberos Authentication

In environments using Active Directory as an identity provider, the Keyfactor Command Management Portal uses integrated Windows authentication by default. Integrated authentication consists of both NTLM and Kerberos authentication types. In some environments, NTLM will work for integrated authentication and users will be able to open the Keyfactor Command Management Portal without further configuration, though not all aspects of the portal support NTLM, including the dashboard and enrollment. In other environments, NTLM will not work at all for the Management Portal, so only Kerberos will be supported. Further configuration is required to make Kerberos authentication work correctly. Even if NTLM is supported and you don't intend to use the portions of the Management Portal that don't work with NTLM, Kerberos is generally preferred for best security practice with Active Directory.

Common scenarios in which NTLM will not work are multi-domain forests and authentication attempts between domains and servers that support only NTLMv2 using clients attempting NTLM.

Configuring the environment to support Kerberos includes these topics:

- Configure browsers to support Integrated Windows Authentication (see Configure Browsers for Integrated Windows Authentication below)
- Configure the service principal name (SPN) for the Keyfactor Command server (see Configure the Service Principal Name for the Keyfactor Command Server on page 2824)
- Configure Kerberos constrained delegation (see Configure Kerberos Constrained Delegation (Optional) on page 2824)

Note: Basic authentication can be used with Active Directory instead of integrated Windows authentication.

Configure Browsers for Integrated Windows Authentication

To support integrated Windows authentication using either NTLM or Kerberos in environments using Active Directory as an identity provider, the browser must be configured correctly to support this integration. This becomes particularly important when only Kerberos is used, as the browser won't allow the user to continue if Kerberos authentication fails, whereas with NTLM authentication, the integration won't work (the user will be prompted to enter a password), but the user will be allowed to continue to the Keyfactor Command Management Portal. Many modern browsers support integrated authentication. The following instructions cover adding the Keyfactor Command server to Window's trusted sites to support integrated authentication for Microsoft Edge and Google Chrome. Configuring Firefox to support integrated authentication is beyond the scope of this guide.

Important: Internet Explorer is no longer supported for Keyfactor Command. For a list of supported browsers, see System Requirements on page 2702.

To configure Windows to support integrated authentication:

1. In Windows either do a search for Internet Options or open Control Panel or Settings and locate Internet Options.
2. In Internet Options, go to the Security tab.
3. On the Security tab, highlight **Local intranet** and click **Sites**.
4. On the Local intranet sites popup, click **Advanced**.
5. On the Local intranet dialog, enter the fully qualified domain name of your Keyfactor Command server and click **Add**.
6. Click **Close** and **OK** until you have closed all the dialogs.
7. Exit your browser (this setting applies to Microsoft Edge and Google Chrome) and open it again to attempt your authentication.

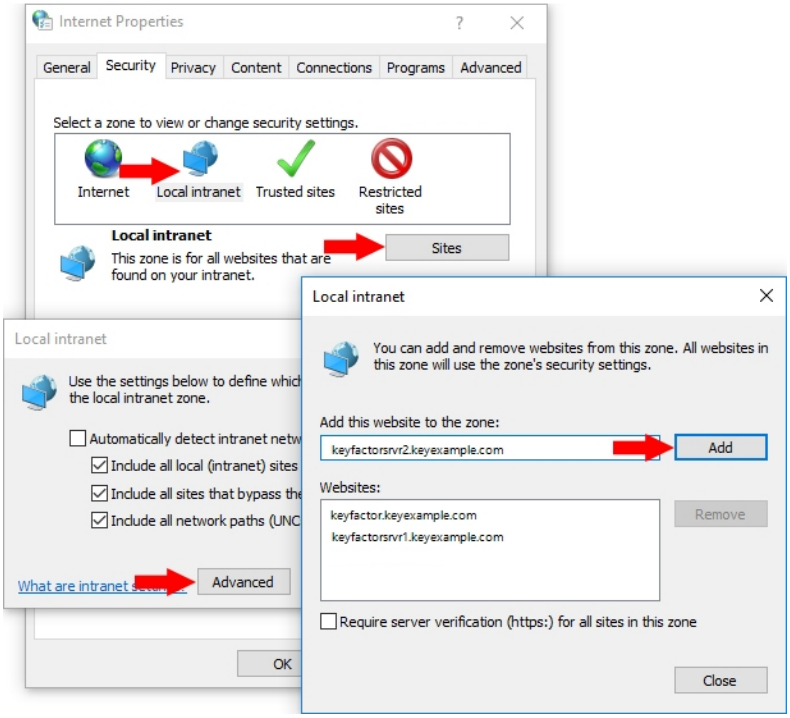


Figure 528: Configure Local Intranet Zone in Internet Properties

Important: It is not sufficient to put the Keyfactor Command server in the **Trusted sites** zone. The server needs to be in the **Local intranet** zone for proper integrated authentication functionality (assuming these zones are still configured as per the default configuration).

Configure the Service Principal Name for the Keyfactor Command Server

In environments using Active Directory as an identity provider, configure the service principal name (SPN) for the Keyfactor Command server as follows:

1. On a server that has the setspn command available (typically it is available on domain controllers, as it installs as part of the Active Directory Domain Services role), open a command prompt using the "Run as administrator" option.

2. Run the following command (where keyfactor.keyexample.com is the fully qualified domain name of your Keyfactor Command server or the DNS alias you are using to reference your Keyfactor Command server, if applicable, and KEYEXAMPLE\svc_keyfactorpool is the domain name and service account name of the service account under which the Keyfactor Command application pool for the Management Portal is running):

   ```
   setspn -s HTTP/keyfactor.keyexample.com KEYEXAMPLE\svc_keyfactorpool
   ```

Configure Kerberos Constrained Delegation (Optional)

If you are using Active Directory as an identity provider and either of these scenarios is true in your environment, you will need to configure Kerberos delegation to the CAs from the Keyfactor Command server hosting the Keyfactor Command Management Portal:

- You wish to use the option in Keyfactor Command to allow interactions with the CA via the Keyfactor Command Management Portal (e.g. certificate approval or revocation) to be done in the context of the user logged into the Keyfactor Command Management Portal rather than in the context of the Keyfactor Command service account under which the application pool is running or an explicit user configured in the CA configuration within Keyfactor Command.
- You wish to enroll for certificates through the Keyfactor Command Management Portal after authenticating to the portal using Kerberos authentication rather than Basic authentication. If you wish to use the Keyfactor Command Management Portal but don't wish to configure delegation or an explicit user configured in the CA configuration within Keyfactor Command, you will need to set the Keyfactor Command Management Portal to support Basic authentication only.

Configuring Kerberos delegation in Active Directory allows the user's Kerberos credentials to be delegated from the Keyfactor Command server to the CA(s) to allow the Keyfactor Command server to act on behalf of the user.

The types of interactions affected by delegation in the Keyfactor Command Management Portal include:

- Enrollment for certificates
- Approval of pending certificate requests
- Denial of pending certificate requests
- Revocation of certificates
- Certificate key recovery

Note: You have the option to turn off delegation for these functions using the *Delegate* settings on each CA configured in Keyfactor Command (see Authorization Methods Tab on page 367 in the *Keyfactor Command Reference Guide*). Delegation is configured separately for management and enrollment functions.

There are two different approaches to configuring constrained delegation:

- With the traditional version of constrained delegation, you configure the service account under which the Keyfactor Command Management Portal application pool runs and the machine account of the Keyfactor Command server to be allowed to delegate **to** each of your CAs.
- With the newer resource-based constrained delegation introduced in Windows server 2012, you configure each of your CAs to be allowed to receive delegation **from** the service account under which the Keyfactor Command Management Portal application pool runs and the machine account of the Keyfactor Command server. This option requires at least one domain controller that's server 2012 or better, though there can be 2008 or 2008 R2 domain controllers in the mix.

With both approaches to constrained delegation, you need to set the service principal name (SPN) for the Keyfactor Command server (see Configure the Service Principal Name for the Keyfactor Command Server on the previous page).

Note: If you're using a Keyfactor CA gateway and the gateway service is running as an Active Directory service account, delegation to that gateway is configured differently than is described below. Refer to the gateway documentation for more information.

Traditional Delegation

Note: Traditional constrained Kerberos delegation across multiple domains is only supported in newer versions of Windows Server for domain controllers. If yours is a multi-domain environment and you cannot locate your CAs following the below instructions, you may need to configure traditional unconstrained Kerberos delegation or configure traditional constrained delegation using ADSIEdit rather than the below method. To configure traditional unconstrained delegation, you would select "Trust this computer for delegation to any service (Kerberos only)" in each of the step 2s, below, and then skip the remainder of the steps in that set of instructions. For assistance configuring traditional constrained delegation using ADSIEdit, contact Keyfactor support (support@keyfactor.com). If none of the traditional constrained delegation methods work in your multi-domain environment, you may need to pursue resource-based constrained delegation instead, which is more forgiving of multi-domain environments.

To configure Kerberos constrained delegation on the machine account of the Keyfactor Command server:

1. Open Active Directory Users and Computers and browse to locate the **machine** account of the Keyfactor Command server and open its properties.

2. On the Delegation tab for the machine account, choose “Trust the computer for delegation to specified services only” and under that “Use any authentication protocol” and then click **Add**.

3. In the Add Services dialog, click **Users or Computers** and browse to locate the computer account for one of the CAs to which you wish to delegate.

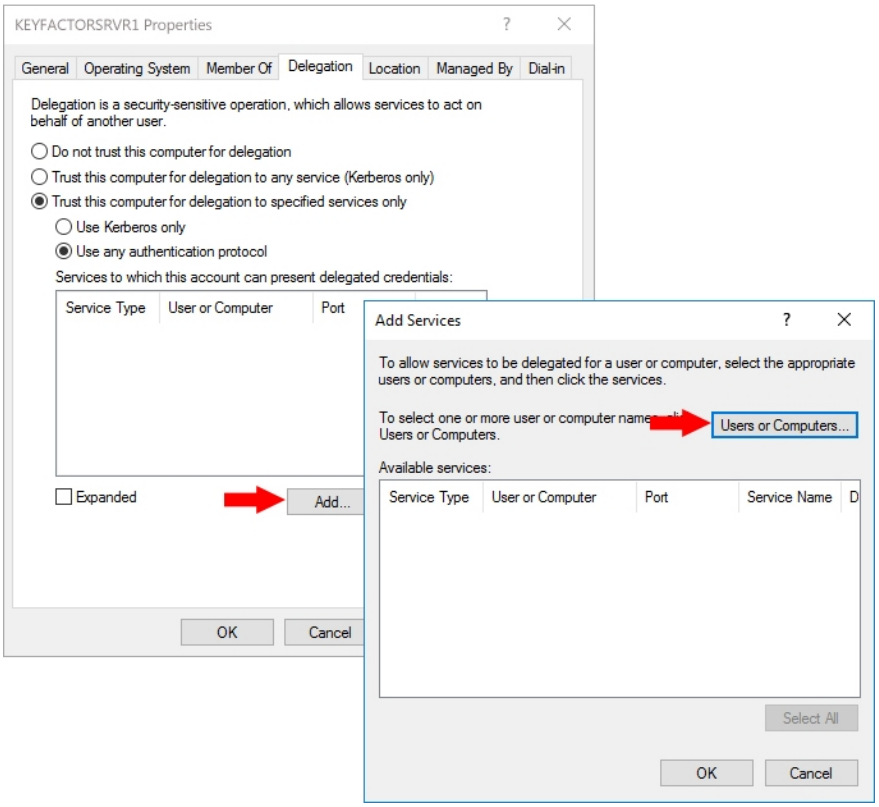


Figure 529: Configure Kerberos Constrained Delegation on the Keyfactor Command Machine Account

4. In the Add Services dialog once the available services have populated, highlight both the **HOST** and the **rpcss** services (hold down the CTRL key when clicking the second service to select both at the same time) and click **OK**.

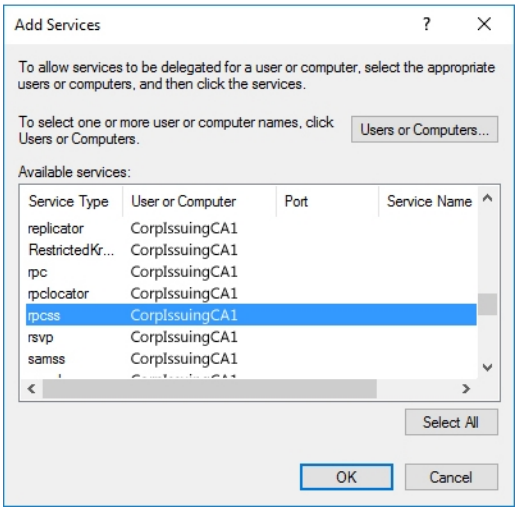


Figure 530: Add HOST and rpcss Service Types for Kerberos Constrained Delegation

5. Back on the Delegation tab of the machine account Properties, you should see the HOST and rpcss services populate the "Services to which this account can present delegated credentials box". The server names do not appear as fully qualified domain names until you close the properties dialog and open it again.

6. Repeat steps 3 and 4 for any other CAs to which you wish to delegate and then click **OK**.

To configure Kerberos constrained delegation on the **service** account under which the Keyfactor Command application pool is running:

1. Open Active Directory Users and Computers and browse to locate the service account under which the Keyfactor Command application pool is running and open its properties.

2. On the Delegation tab for the service account, choose "Trust the computer for delegation to specified services only" and under that "Use Kerberos only" and then click **Add**.

   Important: This is a different configuration setting than for the machine account.

   Tip: The Delegation tab only appears on the properties sheet after you have configured a custom SPN.

3. In the Add Services dialog, click **Users or Computers** and browse to locate the computer account for one of the CAs to which you wish to delegate.

4. In the Add Services dialog once the available services have populated, highlight both the **HOST** and the **rpcss** services (hold down the CTRL key when clicking the second service to select both at the same time) and click **OK**.

5. Back on the Delegation tab of the service account Properties, you should see the HOST and rpcss services populate the “Services to which this account can present delegated credentials box”. The server names do not appear as fully qualified domain names until you close the properties dialog and open it again.

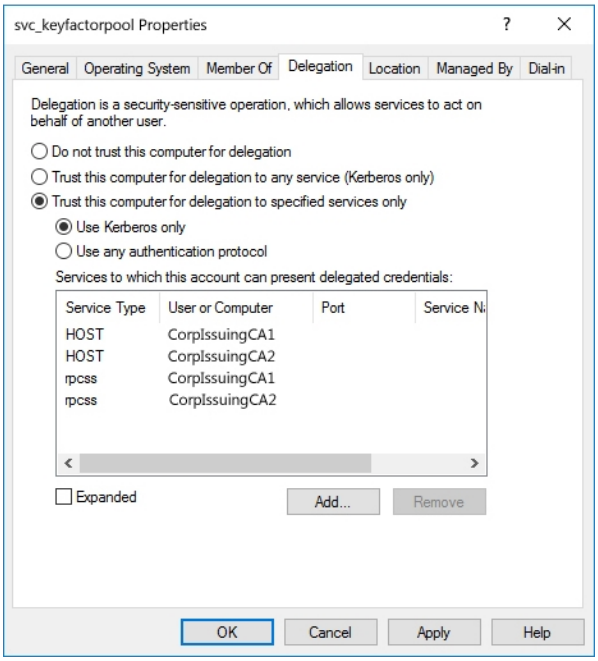


Figure 531: Configure Kerberos Constrained Delegation on the Keyfactor Command Service Account

6. Repeat steps 3 and 4 for any other CAs to which you wish to delegate and then click **OK**.

Resource-Based Delegation

To configure Kerberos resource-based constrained delegation:

1. Create at least one Active Directory security group that will be used for granting delegation permission.

   Tip: The type of group to use and domain to place it in will vary depending on your forest structure and the location of the accounts and resources in the forest. If your Keyfactor Command server(s), CA(s) and application pool service account are all in the same domain, any type of group in that same domain will do. If your Keyfactor Command server (s), CA(s) and/or application pool service account are separated across multiple domains, it becomes more complicated. If you add one or more cross-forest trusts into the mix that you want to do delegation across, that adds another level of complexity. Universal groups cannot be used in a cross-forest scenario, as they are not supported cross-forest. Some possible scenarios include:

- All components (Keyfactor Command server(s), CA(s) and application pool service account) in the same domain: Create a group of any type in the same domain as these components.
- Keyfactor Command server(s) and application pool service account in child domain A and all CAs in the parent domain or child domain B with no cross-forest involvement: Create a universal group in the domain with the CAs (the parent domain or child domain B).
- Keyfactor Command server(s), application pool service account and CA(s) for forest A in the same domain, CAs in a single domain in forest B, forests A and B in a two way trust: Create a group of any type in the forest A domain where the Keyfactor Command server(s) reside and a group of type *domain local* in the forest B domain where the CAs reside; follow the below instructions for both domains and groups.

2. Add the service account under which the Keyfactor Command Management Portal application pool runs to this new security group.

3. Add the machine account for the Keyfactor Command server to this new security group. Repeat for additional Keyfactor Command servers.

4. On an Active Directory domain controller running Windows Server 2012 or better, open a PowerShell window using the "Run as administrator" option. If you're in a multi-domain or cross-forest environment, use a domain controller in the resource domain where the CAs exist.

5. In the PowerShell window, run the following commands, where *KerberosDelegationGroup* is the name of your group for Kerberos delegation and *IssuingCA* is the machine name (no trailing $) of the CA you wish to delegate to:

```
$mygroup = Get-ADGroup -Identity KerberosDelegationGroup
Set-ADComputer IssuingCA -PrincipalsAllowedToDelegateToAccount $mygroup
```

6. Repeat the Set-ADComputer step for any additional CAs.

7. In the PowerShell window, run the following command for each CA to confirm that the group has been associated with the PrincipalsAllowedToDelegateToAccount property on the CA account:

```
Get-ADComputer IssuingCA -Properties PrincipalsAllowedToDelegateToAccount
```

4.4.4.2 Configure Logging

Keyfactor Command provides extensive logging for visibility and troubleshooting. By default, Keyfactor Command places its log files in the C:\Keyfactor\logs directory, generates logs at the *Info* logging level and stores the primary logs for two days before deleting them. If you wish to change these defaults you can open the configuration file for each type of log on each Keyfactor Command server where you wish to adjust logging, and edit the file in a text editor (e.g. Notepad) using the "Run as administrator" option. Each component has its own NLog configuration file and NLog logging output path.

For more information, see Editing NLog on page 796.

4.4.4.3 Configure CA Certificate Synchronization

The Keyfactor Command certificate management, notification and reporting features make use of a SQL database containing certificates from many locations, including:

- Certificates synchronized from Microsoft or EJBCA CAs managed by Keyfactor
- Certificates synchronized from domain-joined Microsoft CAs in your primary forest and forests with which the forest shares a trust
- Certificates synchronized from non-domain-joined EJBCA and Microsoft CAs
- Certificates synchronized from your domain-joined Microsoft CAs in non-trusted forests
- Certificates automatically imported based on SSL synchronization locations
- Certificates imported via Keyfactor CA Gateways from locations such as Entrust and Symantec clouds
- Manually imported certificates
- Certificates inventoried from certificate stores using Keyfactor Command Orchestrators

In order to get these certificates into the Keyfactor Command database so that you can begin using the management, notification and reporting features, you need to configure—at a minimum—CA synchronization. For more information:

- See Certificate Authorities on page 349 for information on configuring CA synchronization for your Microsoft and EJBCA CAs.
- See SSL Discovery on page 453 for information on configuring SSL discovery and monitoring.
- See the separate documentation for each type of CA gateway you have along with Certificate Authorities on page 349 for information on configuring CA synchronization for your CA gateways.
- See Add Certificate on page 74 for information on manually importing a certificate.
- See Installing Orchestrators on page 2875 and Orchestrators on page 481 and Certificate Stores on page 408 for information on inventorying certificates from certificate stores.

For information on using the Keyfactor Command Management Portal, see Using the Management Portal on page 2.

Acquire a Client Certificate for EJBCA CA Authentication

Keyfactor Command uses a client certificate to authenticate to the EJBCA certificate authority to support certificate synchronization, enrollment, and revocation. The certificate that Keyfactor Command uses for authentication needs:

- An extended key usage (EKU) of Client Authentication
- A key usage that includes Digital Signature

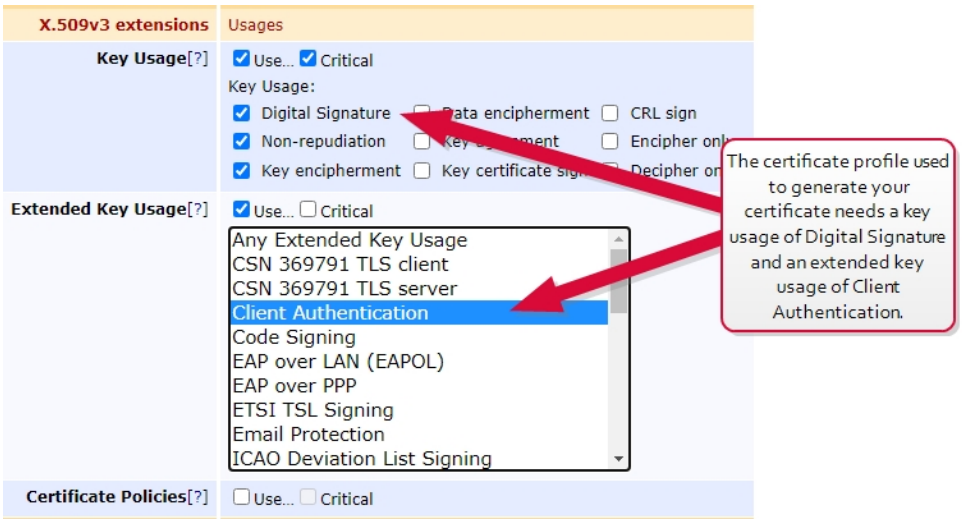


Figure 532: Certificate Profile for EJBCA Client Certificate

The certificate needs to be available as a PKCS#12 (*.pfx) file in order to import it into Keyfactor Command.

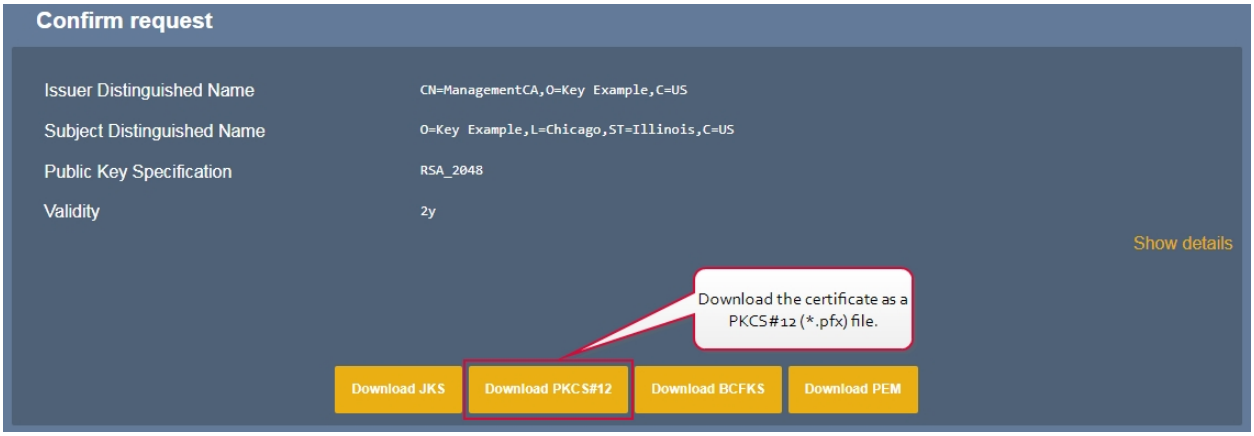


Figure 533: Certificate Download for EJBCA Client Certificate

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

The certificate needs to be granted appropriate access to the EJBCA CA to allow Keyfactor Command interactions with the CA to take place (see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on the next page).

Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs

In order for Keyfactor Command to be able to synchronize certificates from the CAs to the Keyfactor Command database, the service account under which Keyfactor Command makes a connection to the CA must have permissions to *read* the CA databases. For full Keyfactor Command functionality, additional permissions are needed. The permissions needed vary depending on the type of CA and the type of authorization you intend to configure to allow Keyfactor Command and users in Keyfactor Command to interact with the CA.

Microsoft CAs

When you configure Keyfactor Command access to a Microsoft CA, you have the option to enable the *Use Explicit Credentials* option. When this option is enabled, you enter a set of credentials that will be used specifically to access that Microsoft CA, and all management and enrollment tasks for that CA are done in the context of that service account. If you do not enable the *Use Explicit Credentials* option, management tasks (e.g. revocation, certificate synchronization) and enrollments are done in the context of the service account(s) you configure for the Keyfactor Command Service and Keyfactor API the application pool for Keyfactor Command (which are the same service account in many implementations) and individual users. The exact combination of what happens in the context of who depends on the configuration of the delegation options (*Delegate Management Operations* and *Delegate Enrollment*) on the CA when the *Use Explicit Credentials* option is not enabled. Delegation is supported for Basic and Kerberos authentication (see Configure Kerberos Constrained Delegation (Optional) on page 2824) but not NTLM or Token authentication. Use of explicit credentials is mutually exclusive of delegation.

The users and service account(s) you will be using to connect to your Microsoft CA(s) from Keyfactor Command need some set of the following permissions on the CA, based on the configuration of authorization for the CA:

- Read
  To support CA synchronization
- Issue and Manage Certificates
  To support certificate revocation and key recovery
- Manage CA
  To support CRL publication following revocation
- Request Certificates
  To support certificate enrollment through Keyfactor Command

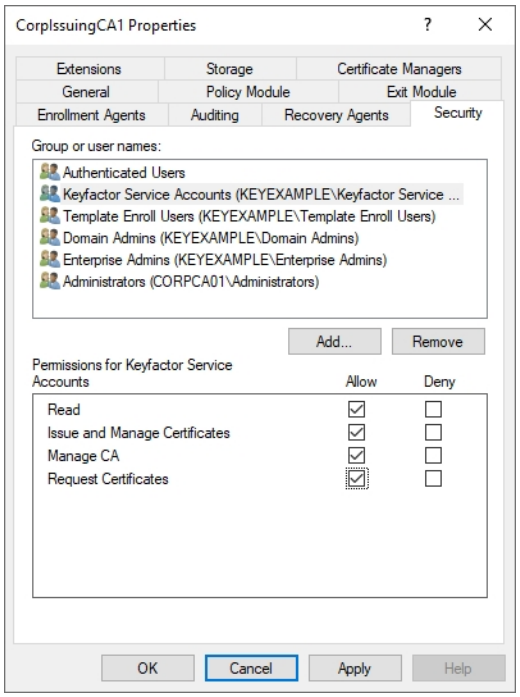


Figure 534: Microsoft CA Permissions

Table 853: Microsoft CA Permission Matrix provides information on what permissions are required on the Microsoft CA based on possible authorization configurations.

In the management console for each CA that Keyfactor Command will be interacting with, open the properties for the CA and grant the users and service account(s) for Keyfactor Command the appropriate permissions for your environment before continuing.

Tip: In order to support PFX and CSR enrollment through the Management Portal, the user initiating the enrollment in the Management Portal must be granted "Request Certificates" permission in the CA if enrollment delegation is enabled. In many environments, all Authenticated Users are granted this permission, allowing the Management Portal users to inherit the permission.

Table 853: Microsoft CA Permission Matrix

	✔ **Use Explicit Credentials**	✘ Use Explicit Credentials ✔ Delegate Management ✔ Delegate Enrollment	✘ Use Explicit Credentials ✘ Delegate Management ✔ Delegate Enrollment	✘ Use Explicit Credentials ✔ Delegate Management ✘ Delegate Enrollment	✘ Use Explicit Credentials ✘ Delegate Management ✘ Delegate Enrollment
Explicit CA-Specific User	Read Issue & Manage Certificates Manage CA Request Certificates	n/a	n/a	n/a	n/a
Keyfactor Command Service Account	None	Read Request Certificates[1]	Read Request Certificates[2]	Read Request Certificates[3]	Read Request Certificates[4]
Keyfactor API Application Pool Account **Note:** A separate	None	Read Issue & Manage Certificates Manage CA Request Certificates[5]	Read Issue & Manage Certificates Manage CA Request Certificates[6]	Read Manage CA Request Certificates	Read Issue & Manage Certificates Manage CA Request Certificates

[1]To support certificate enrollment through Keyfactor Command that occurs in an automated fashion (e.g. certificate renewals initiated by the expiration alert renewal handler).
[2]To support certificate enrollment through Keyfactor Command that occurs in an automated fashion (e.g. certificate renewals initiated by the expiration alert renewal handler).
[3]To support certificate enrollment through Keyfactor Command that occurs in an automated fashion (e.g. certificate renewals initiated by the expiration alert renewal handler).
[4]To support certificate enrollment through Keyfactor Command that occurs in an automated fashion (e.g. certificate renewals initiated by the expiration alert renewal handler).
[5]To support tests of certificate enrollment through Keyfactor Command that occurs in an automated fashion (e.g. certificate renewals initiated by the expiration alert renewal handler).
[6]To support tests of certificate enrollment through Keyfactor Command that occurs in an automated fashion (e.g. certificate renewals initiated by the expiration alert renewal handler).

	✔ **Use Explicit Credentials**	✘ Use Explicit Credentials ✔ Delegate Management ✔ Delegate Enrollment	✘ Use Explicit Credentials ✘ Delegate Management ✔ Delegate Enrollment	✘ Use Explicit Credentials ✔ Delegate Management ✘ Delegate Enrollment	✘ Use Explicit Credentials ✘ Delegate Management ✘ Delegate Enrollment
application pool is required for each virtual directory that will be created for Keyfactor Command in IIS (see Application Pools Tab on page 2786). Requests are made in the context of the user running the application pool for the					

	✔ **Use Explicit Credentials**	✘ Use Explicit Credentials ✔ Delegate Manage-ment ✔ Delegate Enrollment	✘ Use Explicit Credentials ✘ Delegate Manage-ment ✔ Delegate Enrollment	✘ Use Explicit Credentials ✔ Delegate Manage-ment ✘ Delegate Enrollment	✘ Use Explicit Credentials ✘ Delegate Manage-ment ✘ Delegate Enrollment
Keyfacto-r API virtual director-y.					
Individual Users	None	Read Issue & Manage Certi-ficates Request Certi-ficates	Read Request Certificates	Read Issue & Manage Certi-ficates	None

EJBCA CAs

Management (e.g. revocation, certificate synchronization) and enrollment requests to an EJBCA CA are made in the context of the end entity associated with the client certificate selected in each CA configuration in the Keyfactor Command Management Portal to provide authentication to the EJBCA CA (see Acquire a Client Certificate for EJBCA CA Authentication on page 2830). The access rule created or used for this needs to grant sufficient permissions to allow the end entity to synchronize certificates. For full functionality, it needs the following permissions:

- /administrator/

  To support Keyfactor Command making API requests to the EJBCA CA
- /ca/*[your_ca_name]*/

  To support Keyfactor Command access to your CA
- /ca_functionality/create_certificate/

  To support certificate enrollment through Keyfactor Command
- /ca_functionality/create_crl/

  To support CRL publication following revocation
- /ca_functionality/view_ca/

To support retrieval of CA information

- /ca_functionality/view_certificate/

  To support CA synchronization

- /ca_functionality/view_certificate_profiles/

  To support template import

- /endentityprofilesrules/*[your_end_entity_profile_name]*/create_end_entity/

  To support creation of end entities (a new end entity is created for each Keyfactor Command certificate enrollment unless the *Enforce Unique DN* option is enabled and the new certificate's DN matches that of an existing certificate)

- /endentityprofilesrules/*[your_end_entity_profile_name]*/edit_end_entity/

  To support certificate enrollment with the *Enforce Unique DN* option through Keyfactor Command and certificate renewal through Keyfactor Command

- /endentityprofilesrules/*[your_end_entity_profile_name]*/revoke_end_entity/

  To support certificate revocation through Keyfactor Command

- /endentityprofilesrules/*[your_end_entity_profile_name]*/view_end_entity/

  To support certificate enrollment through Keyfactor Command

- /ra_functionality/create_end_entity

  To support creation of end entities (a new end entity is created for each Keyfactor Command certificate enrollment unless the *Enforce Unique DN* option is enabled and the new certificate's DN matches that of an existing certificate)

- /ra_functionality/edit_end_entity

  To support certificate enrollment with the *Enforce Unique DN* option through Keyfactor Command and certificate renewal through Keyfactor Command

- /ra_functionality/revoke_end_entity

  To support certificate revocation through Keyfactor Command

- /ra_functionality/view_end_entity

  To support certificate enrollment through Keyfactor Command

- /system_functionality/view_administrator_privileges

  To support overall functionality

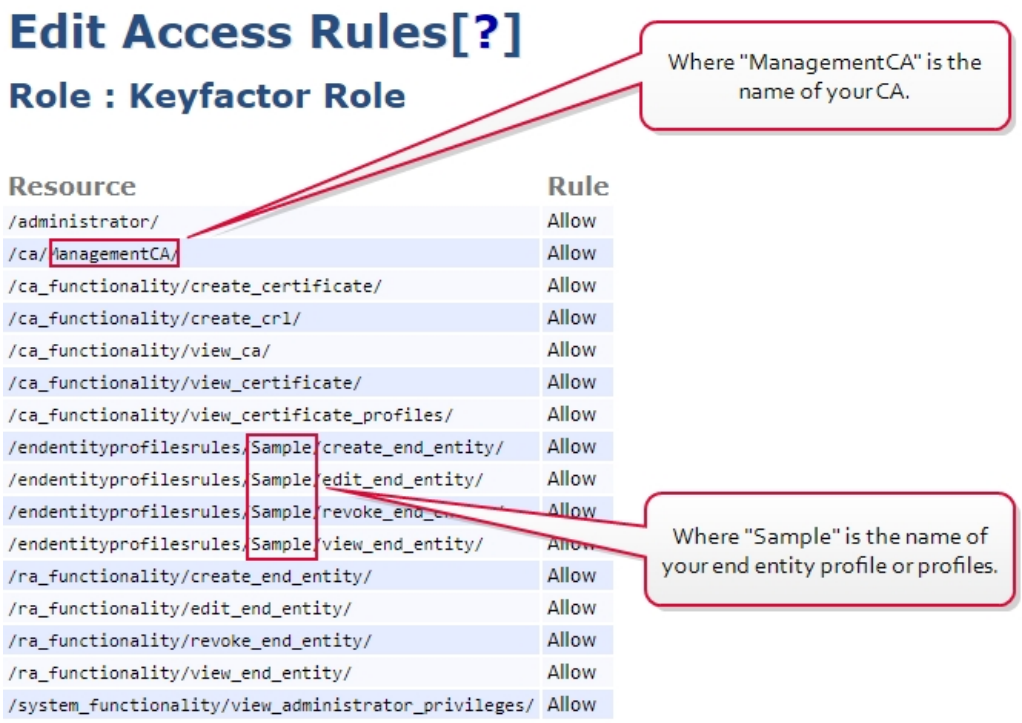


Figure 535: EJBCA Access Permissions

You may either create a new access rule that limits access to just these required permissions, or use an existing access rule. In either case, you need to add the certificate used to authenticate Keyfactor Command to the EJBCA CA as a member of that access rule.

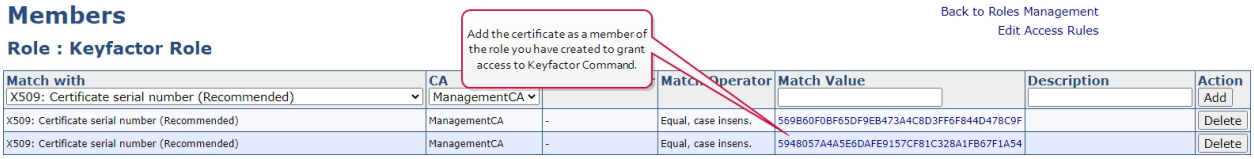


Figure 536: Add Client Certificate as Member of EJBCA Access Rule

Enable and Start the Keyfactor Command Service

The Keyfactor Command Service runs on the Keyfactor Command server hosting the Services role and controls database synchronization, among other jobs. During the Keyfactor Command configuration process you configured the service account under which the Keyfactor Command Service will run and may have configured the service to start automatically at server boot time (see Configure: Service on page 2802).

Tip: The Keyfactor Command Service can be installed on every server that Keyfactor Command is installed on—for instance in a high availability scenario. This allows the service to check out jobs via a locking mechanism that enforces that any jobs are running on only one

service at a time. There is a timeout setting for the service locking mechanism that may be adjusted if needed. [1]

To start the service (if it hasn't started automatically):

1. On the Keyfactor Command server hosting the Services role, open the Services MMC.
2. In the Services MMC confirm that the Keyfactor Command Service is set to a Startup Type of Automatic (if desired). If the service is not running, click the green arrow to start it.

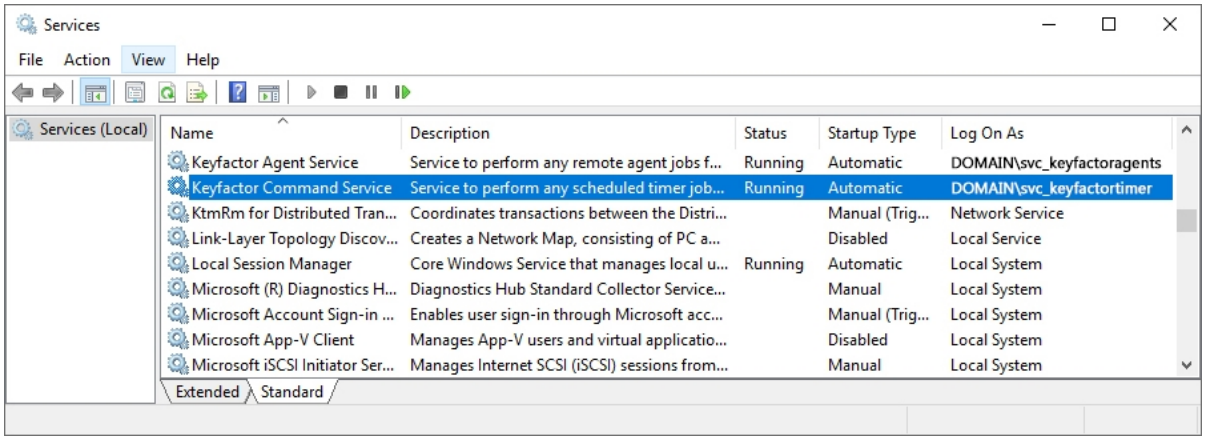


Figure 537: Keyfactor Command Service

If you have configured a synchronization schedule for your CA (see Configure CA Certificate Synchronization on page 2830), the CA(s) will begin to synchronize when the first scheduled scan time is reached. Scans scheduled at intervals match to clock times, so a scan set at an interval of 15 minutes will run at 6:00, 6:15, 6:30, 6:45, etc. You can check the Keyfactor Command timer service log file on the Keyfactor Command Services server to confirm that synchronization is operating as expected. You can also use the Certificate Search feature in the Keyfactor Command Management Portal to confirm the certificates are appearing in the Keyfactor Command database. The database synchronization begins with the oldest certificates in the CA database, which may be expired or revoked. Be sure to toggle the *Include Revoked* and *Include Expired* options, see Include Expired and Revoked Certificates in Certificate Search on the next page, when checking to see if synchronization is working. See Certificate Search Page on page 34 for information on using the search.

---

[1] Adjust the timeout setting for the service locking mechanism:

1. Navigate to the Configuration folder (default location: *C:\Program Files\Keyfactor\Keyfactor Platform\Configuration*)
2. Open the file: **ConfigurationWizardConsole.exe.config**
3. Edit the value on the line: <add key="Keyfactor.Sql.DbCommandTimeout" value="1800" />

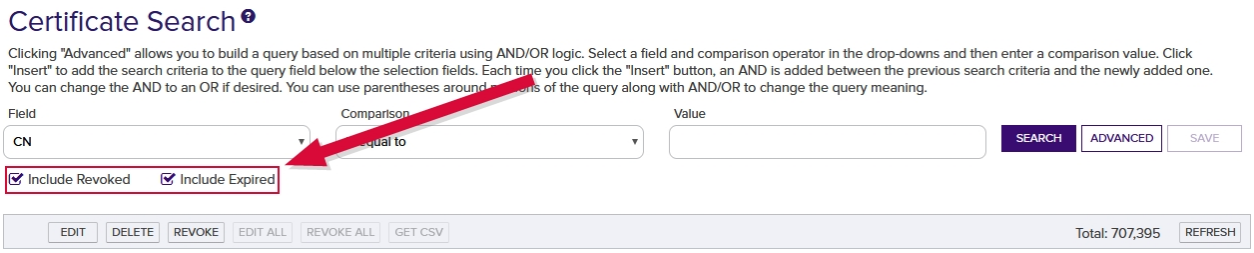


Figure 538: Include Expired and Revoked Certificates in Certificate Search

4.4.4.4 Create or Identify Certificate Templates for Enrollment

This step only needs to be completed if your Keyfactor Command license includes certificate enrollment and you plan to use this feature.

Note: Keyfactor Command and this documentation use the term *template* generically to refer to Microsoft certificate templates and EJBCA certificate templates. EJBCA templates are built from the EJBCA end entity profile and certificate profile and named using a naming scheme of *<end entity profile name>_<certificate profile name>* and *<end entity profile name>* (*<certificate profile name>*) for the template name and template display name.

The enrollment function in the Keyfactor Command Management Portal is generally used by administrators to request certificates for use on servers, network devices, and similar equipment. There's a good chance that certificate templates for these purposes already exist in your environment. To prepare for the Keyfactor Command installation, you need to gather a list of the CAs that will be used to issue certificates through the Keyfactor Command Management Portal and a list of the *template names* (vs template display names) of the templates that will be used for this (Microsoft CAs) or certificate profiles and end entity profiles (EJBCA CAs). If any new templates or profiles need to be created for this purpose, they should be created before completing the Keyfactor Command post-installation steps.

For Microsoft CAs, the security settings on your existing templates may need to be modified to allow users to enroll for certificates using them through the Keyfactor Command Management Portal, depending on how the templates have been used previously. For CAs in the local forest (the forest in which Keyfactor Command is installed) and forests in a two-way trust with the local forest, enrollment through the Keyfactor Command Management Portal is often done in the context of the user logged into the portal. This differs from enrolling for a certificate through the Microsoft certificates MMC, where requests for computer certificates (such as web server certificates) are done in the context of the machine account from which the certificate is requested, not the user account, and thus the machine account needs permissions, not the user. When using the Keyfactor Command Management Portal, each of the users who will use one of the enrollment functions needs **Read** and **Enroll** permissions on the templates they will be using through the portal (see Grant the Keyfactor Command Users and Service Account(s) Permissions on the CAs on page 2832 for more details).

Tip: Enrollment through the Keyfactor Command Management Portal against remote Microsoft CAs (CAs in a forest with either no trust with the forest in which Keyfactor

Command is installed or a one-way trust) are done in the context of the service account configured on the Management Portal CA record for *Explicit Credentials* (see Create Service Accounts for Keyfactor Command on page 2757).

The Keyfactor Command Management Portal offers the option of using a different set of CAs and templates for each of the two different enrollment methods—PFX and CSR. As you collect your list of CAs and templates, you will need to decide whether you want to use the same CAs and templates for both types of enrollment or whether each type of enrollment will have a unique list of CAs and templates.

The list of templates used for enrollment in the Keyfactor Command Management Portal is configured through the Keyfactor Command Management Portal template management. Although in previous releases of Keyfactor Command, the templates and CAs for enrollment were configured during installation, this is now done as a post-install step in the Management Portal. See Certificate Authorities on page 349 and Certificate Template Operations on page 381.

4.4.4.5 Configure Renewal Handler Permission

The expiration renewal event handler allows you to execute a certificate renewal automatically for each expiring certificate that is found in a supported certificate store for each expiration alert when the alert task is triggered by the execution of the expiration alerts. In order for the renewal handler to execute successfully, the Active Directory service account under which the Keyfactor Command Service runs must have select permissions in the Keyfactor Command Management Portal. In addition, if you wish to test the execution of expiration alerts with renewal handlers and your IIS application pool runs in the context of a different Active Directory service account than the Keyfactor Command Service, the Active Directory service account for the Keyfactor API IIS application pool must also be granted these permissions.

Note: If your Microsoft CA has been configured with the *Use Explicit Credentials* option, the permissions described here need to be granted to the user specified by the *Use Explicit Credentials* option, not either of the above-referenced service accounts.
If you're using an EJBCA CA, no further permissions need to be granted and this step may be skipped.

If you don't plan to use the expiration renewal handler, you can skip this step.

To configure permissions for the service account(s) to support use of the expiration renewal handler:

1. In the Keyfactor Command Management Portal, browse to *System Settings Icon* > *Security Roles & Identities*.
2. On the Security Roles and Identities page on the Security Roles tab, click **Add** to create a new role to be used just to grant permissions to the service account(s) to support use of the expiration renewal handler.
3. On the Details tab, give it an appropriate name and description to reflect this usage.

4. On the Global Permissions tab:

    a. Select *Certificate Enrollment* and click the **Enroll PFX** toggle to enable it.

    b. Select *Certificate Store Management* and click the **Read** and **Schedule** toggles to enable them.

    c. Select *Certificates* and click the **Read** toggle to enable it.

    d. Select *Management Portal* and click the **Read** toggle to disable it, if enabled.

5. Click **Save** to save the role.

6. On the Security Roles and Identities page on the Security Identities tab, click **Add** to add a new security identity.

7. In the Security Identities dialog, enter the Active Directory user name of the service account under which the Keyfactor Command Service runs using DOMAIN\username format and click **Save**. If the account resolves correctly, the new identity will be saved and the dialog will close.

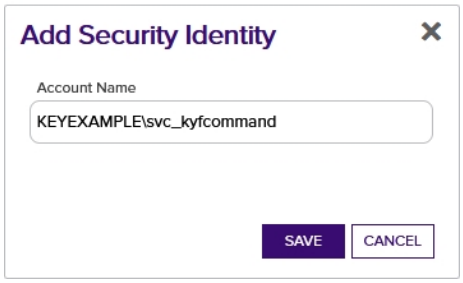


Figure 539: Configure Expiration Renewal Handler: Add New Identity

8. If your IIS application pool runs as a different Active Directory service account from that used for the Keyfactor Command Service, repeat steps six and seven for the IIS application pool service account.

9. In the Security Identity Editor section of the page, double-click the Keyfactor Command Service identity in the identity grid, right-click the row in the identity grid and choose **Edit Roles** from the right-click menu, or highlight the Keyfactor Command Service identity in the identity grid and click **Edit Roles** at the top of the identity grid.

10. In the Roles dialog, select the role you created for the expiration renewal handler in the Available Roles list and use the right arrow to move the role to the Current Roles list. Click **Save** to assign the role to the identity.

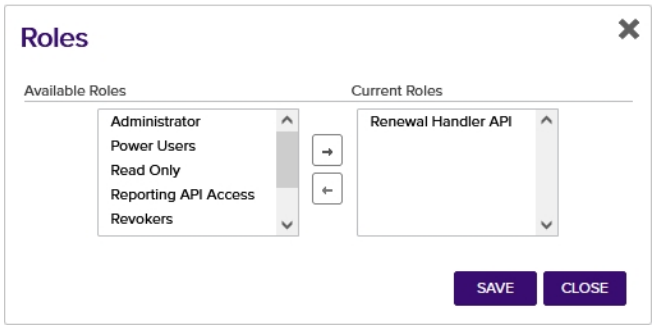


Figure 540: Configure Expiration Renewal Handler: Assign Role to Identity

11. If your IIS application pool runs as a different Active Directory service account from that used for the Keyfactor Command Service, repeat steps nine and ten for the IIS application pool service account.

4.4.4.6 Create a Certificate Template for Mac Auto-Enrollment

This step only needs to be completed if your Keyfactor Command license includes Mac auto-enrollment and you plan to use this feature.

To create the certificate template that will be used for Mac auto-enrollment:

1. On the CA that will issue the Mac auto-enrollment certificates, open the Certification Authority management tool.

2. In the Certification Authority management tool, drill down to locate the Certificate Templates folder. Right-click the **Certificate Templates** folder and choose **Manage**. This will open the Certificate Templates Console.

3. In the Certificate Templates Console, right-click the User template and choose **Duplicate Template**.

4. If prompted with a Duplicate Template dialog (some versions of Windows), choose Windows Server 2003 Enterprise and click **OK**.

5. General Tab: In the Properties of New Template dialog on the General tab, enter **Mac Auto-Enrollment** (or an alternate name of your choosing) in the **Template display name** field. The **Template name** will be auto-populated based on the text you enter in the **Template display name**. Select a **Validity period** for the certificate that's appropriate for your environment.

6. Extensions Tab: If you plan to use the certificates to authenticate to enterprise systems, you will need to ensure that **Client Authentication** is set as the only application policy in the certificate. To do this, in the **Extensions included in this template** section of the Extensions tab, highlight **Application Policies** and click the **Edit...** button. In the Edit Application Policies Extensions dialog, remove the **Encrypting File System** and **Secure Email** policies and click **OK**.

7. Security Tab: In the Properties of New Template dialog on the Security tab, add the Active Directory group of users who will be allowed to auto-enroll from Macs and grant this group **Read**, **Enroll**, and **Autoenroll** permissions on the template.
8. Click **OK** to save the new template.
9. Back in the Certification Authority management tool, right-click the **Certificate Templates** folder and choose **New->Certificate Template to Issue**. Select the **Mac Auto-Enrollment** template from the list presented and click **OK**.

4.5 Keyfactor CA Policy Module

The Keyfactor CA Policy Module includes four certificate authority policy handlers that can be used to alter or restrict the functionality of a Microsoft certificate authority. The policy handlers are installed on the Microsoft CA and enabled through the Microsoft CA properties page. The available policy handlers are:

RFC 2818 Policy Handler

Automate inclusion of a DNS SAN matching the CN of the requested certificate in certificate enrollments for a defined set of CA templates.

SAN Attribute Policy Handler

Allow the addition of SANs not included in the CSR when making a CSR enrollment request. The added SANs will overwrite any existing SANs in the CSR. This functionality is the same as that seen with the Microsoft default policy module for the CA as a whole when the CA EDITF_ATTRIBUTESUBJECTALTNAME2 flag is set except the SAN Attribute Policy Handler provides the ability to control SAN addition on a template-by-template basis without the need to enable the Microsoft CA EDITF_ATTRIBUTESUBJECTALTNAME2 flag.

vSCEP™ Policy Handler

Allow secure control of on-device key generation during certificate enrollment for iOS and Mac devices.

Whitelist Policy Handler

Enforce that certificate requests for a given template or templates can only be initiated from a given computer or set of computers.

Important: If you're upgrading from a previous version of the Keyfactor CA Policy Module, refer to the *Keyfactor Command Upgrade Overview* for important upgrade instructions and a required upgrade script. Newer versions of Keyfactor CA Policy Module **cannot** be installed over the top of existing Keyfactor CA Policy Module installations to complete an upgrade.

4.5.1 System Requirements

The Keyfactor CA Policy Module is supported on Microsoft certificate authorities running on Windows Server 2016 or higher. It interoperates with Keyfactor Command versions 9.0 or greater.

The policy module requires the Microsoft .NET **Desktop** Runtime version 6.0 (x64). Version 6.0 is available for download from Microsoft:

https://dotnet.microsoft.com/download/dotnet/6.0/runtime

At the above link, this would be the **Download x64** option under the "Run desktop apps" heading.

You can use the following PowerShell command to check the .NET core version(s) installed on a server (if any):

```
dotnet --list-runtimes
```

Output from this command will look something like this if you have the correct 6.0 x64 version of the .NET Desktop Runtime installed (notice the paths are in C:\Program Files, not C:\Program Files (x86), indicating this is the x64 version):

```
Microsoft.NETCore.App 6.0.11 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.WindowsDesktop.App 6.0.11 [C:\Program
Files\dotnet\shared\Microsoft.WindowsDesktop.App]
```

The policy module also requires a Keyfactor Command license key for the current release with a policy module license.

Important: If you're upgrading from a previous version of the Keyfactor CA Policy Module, refer to the *Keyfactor Command Upgrade Overview* for important upgrade instructions and a required upgrade script. Newer versions of Keyfactor CA Policy Module **cannot** be installed over the top of existing Keyfactor CA Policy Module installations to complete an upgrade.

4.5.2 Preparing for the Keyfactor CA Policy Module

The preparation steps necessary for the Keyfactor CA Policy Module vary depending on the policy handler(s) you intend to use.

Important: If you're upgrading from a previous version of the Keyfactor CA Policy Module, refer to the *Keyfactor Command Upgrade Overview* for important upgrade instructions and a required upgrade script. Newer versions of Keyfactor CA Policy Module **cannot** be installed over the top of existing Keyfactor CA Policy Module installations to complete an upgrade.

The policy handlers have the following preparation requirements:

RFC 2818 Policy Handler

During the configuration of the RFC 2818 Policy Handler, you will need to define the list of Microsoft certificate templates that will automatically be assigned a DNS SAN matching the certificate's CN when a certificate enrollment request reaches the CA. These templates are configured by selecting them from a list. You will need to have this list of templates ready.

SAN Attribute Policy Handler

During the configuration of the SAN Attribute Policy Handler, you will need to define the list of Microsoft certificate templates that will allow certificate enrollment requests via CSR to submit SANs outside of the CSR for inclusion in the final certificate, replacing any SANs originally in the CSR. These templates are configured by selecting them from a list. You will need to have this list of templates ready.

vSCEP™ Policy Handler

During the configuration of the vSCEP™ Policy Handler, you will need to enter the URL to the vSCEP service on your Keyfactor Command server and the username and password of a service account that the policy handler will use to make requests to the vSCEP API on the Keyfactor Command server. The user you enter here needs to be a member of the group you configure for *Allowed Users/Groups* on the vSCEP Service tab in the Keyfactor Command configuration wizard (see Install the Keyfactor Command Components on the Keyfactor Command Server(s) on page 2780). The service account needs to be created in Active Directory prior to installation of the Keyfactor CA Policy Module software, and the person installing the Keyfactor CA Policy Module software needs to know the service account domain, username and password.

Whitelist Policy Handler

During the configuration of the SAN Attribute Policy Handler, you will need to define the list of Microsoft certificate templates that will be gated by the handler and the list of machines that will be allowed to use these templates. Any templates you include will be available for enrollment only from machines you include in the allowed list. The purpose of this handler is to force all enrollments to be made from the Keyfactor Command server(s), so the list of machines should include your Keyfactor Command server(s). The templates for this policy handler are configured by typing in their certificate *template name* (short name), so you will need an exact list of the template names.

In addition, you will need to have your Keyfactor product license available for upload into the policy module once installed to activate it.

4.5.3 Installing the Keyfactor CA Policy Module Handlers

These steps only need to be completed if your Keyfactor Command license includes the Keyfactor CA Policy Module and you plan to use this feature and one or more of its policy handlers. Review the policy handlers to determine if one or more of them meets a need in your environment.

Important: For a CA Clustered solution, if the Keyfactor CA Policy Module is installed on a node then configured, then failed over to another node, this will corrupt the check point key. The module must be installed on BOTH nodes, configured on one node, then failed over to the other node.

The available policy handlers are:

RFC 2818 Policy Handler

Automate inclusion of a DNS SAN matching the CN of the requested certificate in certificate enrollments for a defined set of CA templates.

SAN Attribute Policy Handler

Allow the addition of SANs not included in the CSR when making a CSR enrollment request. The added SANs will overwrite any existing SANs in the CSR. This functionality is the same as that seen with the Microsoft default policy module for the CA as a whole when the CA EDITF_ATTRIBUTESUBJECTALTNAME2 flag is set except the SAN Attribute Policy Handler provides the ability to control SAN addition on a template-by-template basis without the need to enable the Microsoft CA EDITF_ATTRIBUTESUBJECTALTNAME2 flag.

Important: By default, Microsoft CAs do not support the addition of SANs not included in the CSR when making a request using a CSR enrollment method. To enable your CA to support requesting certificates with additional SANs, you must either install and configure the Keyfactor Command SAN Attribute Policy Handler on the CA(s) or enable the Microsoft CA EDITF_ATTRIBUTESUBJECTALTNAME2 flag. There are security risks inherent in enabling either of these options on your CA. Keyfactor recommends that you do not enable these options unless it is an absolute requirement. With the SAN Attribute Policy Handler, you can limit the risk by limiting the exposure to just selected templates. Keyfactor further recommends that you:

- Use the SAN Attribute Policy Handler only with templates that require CA manager approval so that a manager will be required to review the request and the added SANs before the certificate is issued.
- Use the SAN Attribute Policy Handler in conjunction with the Whitelist Policy Handler to limit requests for the selected templates to being initiated only by the Keyfactor Command server(s).
- Configure server level monitoring with a product such as Microsoft's System Center Operations Manager (SCOM) to provide alerts for any changes relating to the CA(s) configured with the SAN Attribute Policy Handler so that, for example, changes to the templates configured to support SAN addition do not go unnoticed.

vSCEP™ Policy Handler

Allow secure control of on-device key generation during certificate enrollment for iOS and Mac devices.

Whitelist Policy Handler

Enforce that certificate requests for a given template or templates can only be initiated from a given computer or set of computers.

Note: The following Windows update affects how certificate requests are built when sent to a Microsoft CA and may cause enrollments done outside Keyfactor Command against a Microsoft CA configured with the Whitelist Policy Handler to fail.

https://support.microsoft.com/en-us/topic/kb5014754-certificate-based-authentication-changes-on-windows-domain-controllers-ad2c23b0-15d8-4340-a468-4d4f3b188f16

The processing order of the handlers currently available in the Keyfactor CA Policy Module, when used together on the same machine, is significant for some handlers and not others. Specifically, the processing order is not significant for the vSCEP™ Policy Handler and Machine Whitelist Policy handler. These handlers may be placed anywhere within the list of handlers. However, the processing order does matter for the SAN Attribute Policy Handler and the RFC 2818 Policy Handler. When these two handlers are used together, the SAN Attribute Policy Handler must be placed on the list above the RFC 2818 Policy Handler to allow the SAN Attribute Policy Handler to be processed before the RFC 2818 Policy Handler. This is because the SAN Attribute Policy Handler removes any existing SANs on the enrollment request and replaces them with those specified in the request outside of the CSR—such as those entered in the optional SAN section on the CSR page of the Keyfactor Command Management Portal. This includes any SANs added by the RFC 2818 Policy Handler.

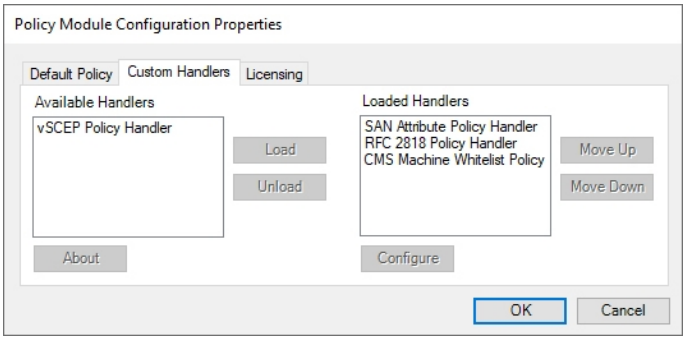


Figure 541: Keyfactor CA Policy Module Policy Module Handler Ordering

When the Keyfactor CA Policy Module is used, the policy module listed on the Default Policy tab of the Policy Module Configuration Properties dialog is run first when a request reaches the CA. This default policy might be the standard Windows default, as shown Figure 542: Default Policy Module, or it might be another non-built-in policy module, such as the Microsoft FIM CM Policy Module. After the default policy module runs, the Loaded Handlers on the Custom Handlers tab of the Policy Module Configuration Properties dialog are run in the order listed (top to bottom). After all the handlers have been run, the result (approved, denied, or marked as pending) is returned to the CA for processing.

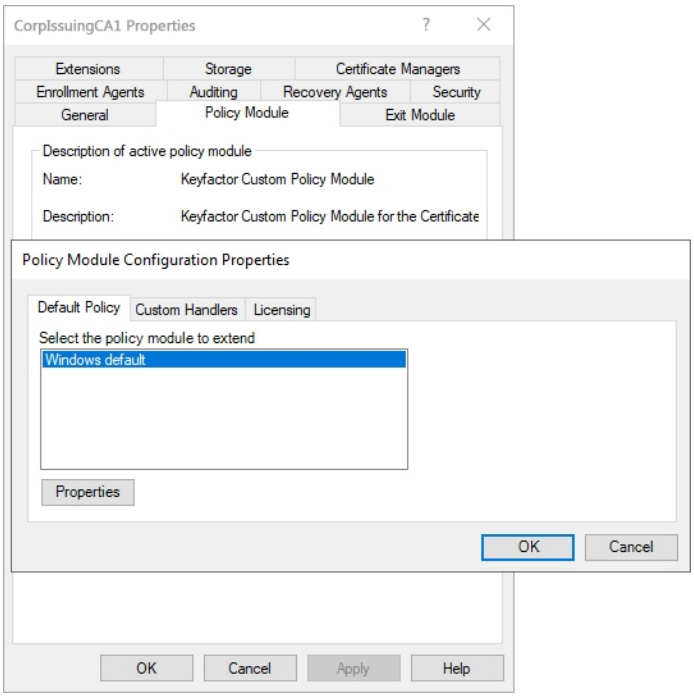


Figure 542: Default Policy Module

Tip: Once the installation is complete, the configuration options for the policy handlers can be found in the registry on the CA in the following paths (where CA_LOGICAL_NAME is the logical name of the local CA):

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\CertSvc\Configuration\CA_
LOGICAL_NAME\PolicyModules\CMS_Custom.Policy\PolicyHandlers\RFC2818.PolicyHandler

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\CertSvc\Configuration\CA_
LOGICAL_NAME\PolicyModules\CMS_
Custom.Policy\PolicyHandlers\SANAttribute.PolicyHandler

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\CertSvc\Configuration\CA_
LOGICAL_NAME\PolicyModules\CMS_Custom.Policy\PolicyHandlers\vSCEP.PolicyHandler

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\CertSvc\Configuration\CA_
LOGICAL_NAME\PolicyModules\CMS_
Custom.Policy\PolicyHandlers\CMSWhitelist.PolicyHandler
```

Important: These registry keys should not be modified without advice from Keyfactor support.

4.5.3.1 Install the Keyfactor RFC 2818 Policy Handler

To begin the RFC 2818 Policy Handler installation, execute the KeyfactorCAModuleInstaller.msi file from the Keyfactor installation media and install as follows.

1. On the first installation page, click **Next** to begin the setup wizard.

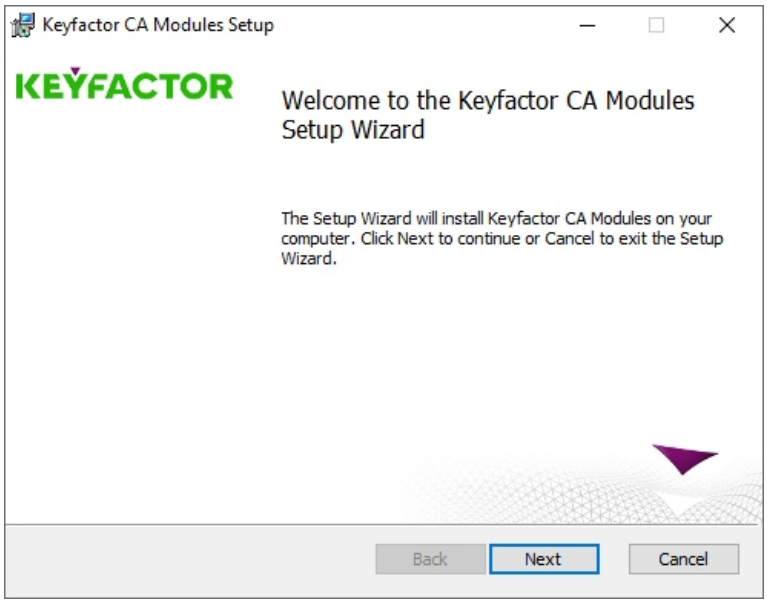


Figure 543: Install RFC 2818 Policy Handler: Begin Setup Wizard

2. On the next page, read and accept the license agreement and click **Next**.

3. On the next page, select the components to install. For the RFC 2818 Policy Handler, deselect all the components except the RFC 2818 Policy Handler component. If desired, you can highlight Keyfactor Custom Policy Module and click **Browse** to select an alternate installation location for the files. The default installation location is:

    C:\Program Files\Keyfactor\Keyfactor CA Modules

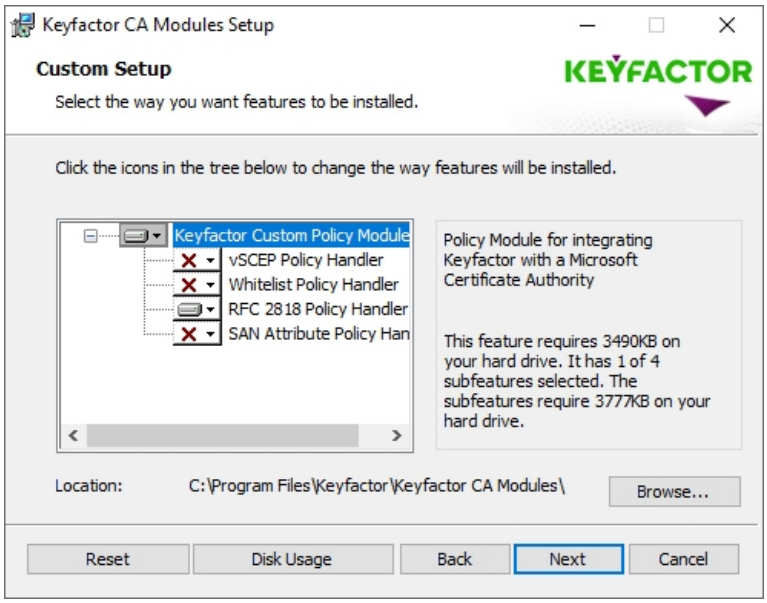


Figure 544: Install RFC 2818 Policy Handler: Select Components

4. On the next screen, click **Install**.

5. On the final installation wizard page, leave the *Launch the CA MMC snap-in now* box selected and click **Finish**. The Microsoft Certification Authority management tool should start automatically. This can take several seconds.

6. In the Certification Authority management tool, right-click the CA name at the top of the tree and choose **Properties**.

7. In the Properties dialog for the CA on the CA Policy Module tab, click **Select**, highlight the **Keyfactor Custom Policy Module** in the Set Active Policy Module dialog and click **OK**.

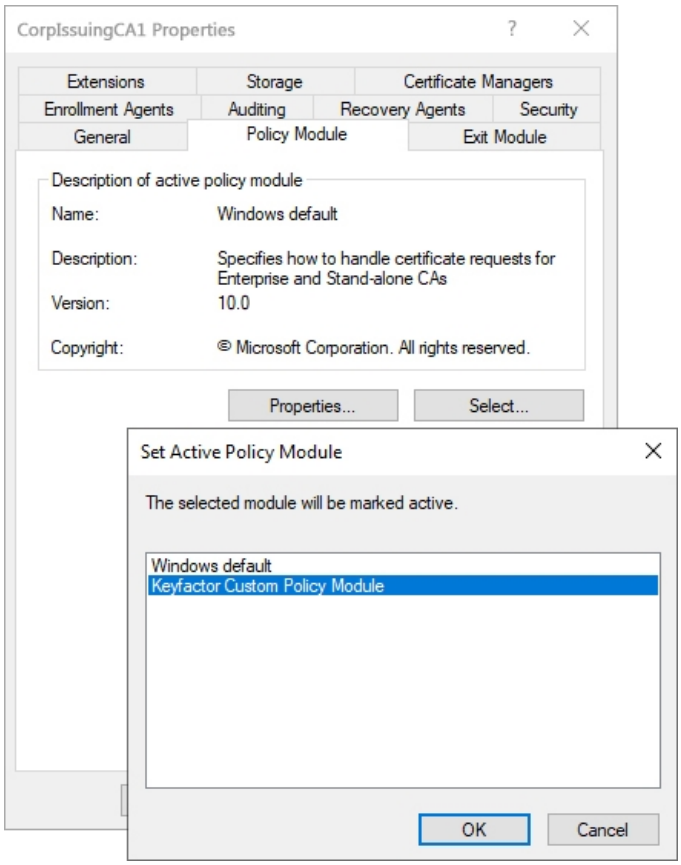


Figure 545: Enable the Keyfactor CA Policy Module

8. In the Properties dialog for the CA on the CA Policy Module tab, click **Properties**.

9. On the Licensing tab of the Policy Module Configuration Properties page, click **Upload License** and browse to locate the license file provided to you by Keyfactor. This file should have the extension CMSLICENSE.

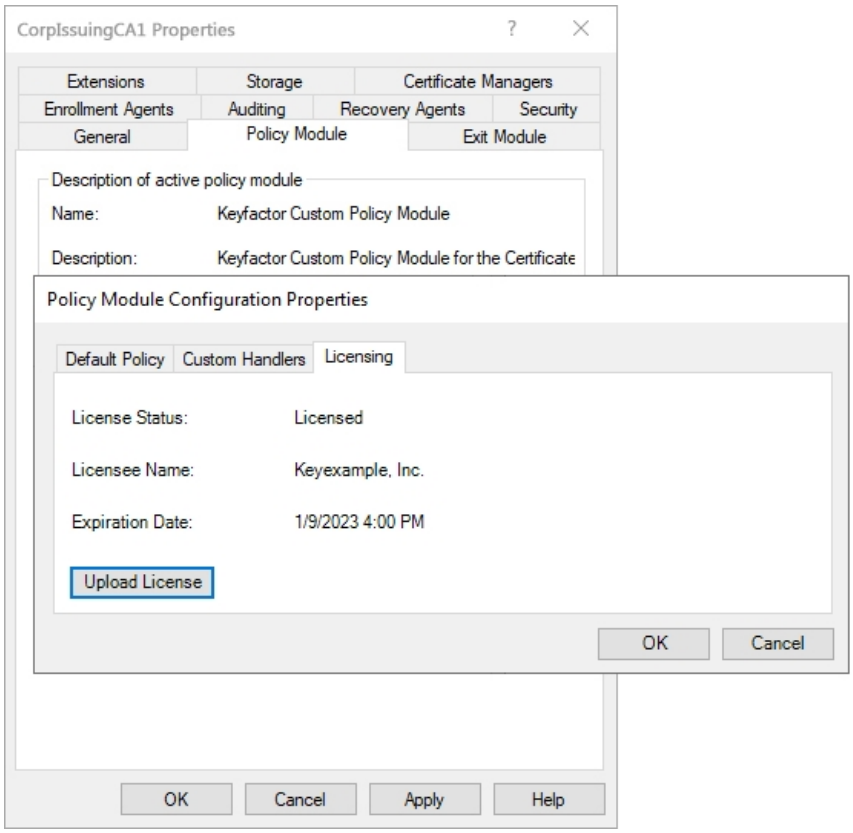


Figure 546: Upload the Keyfactor CA Policy Module License

10. On the Custom Handlers tab of the Policy Module Configuration Properties page, highlight the **RFC 2818 Policy Handler** under Loaded Handlers, click **Load** to move it over to the loaded handlers, and click **OK**.

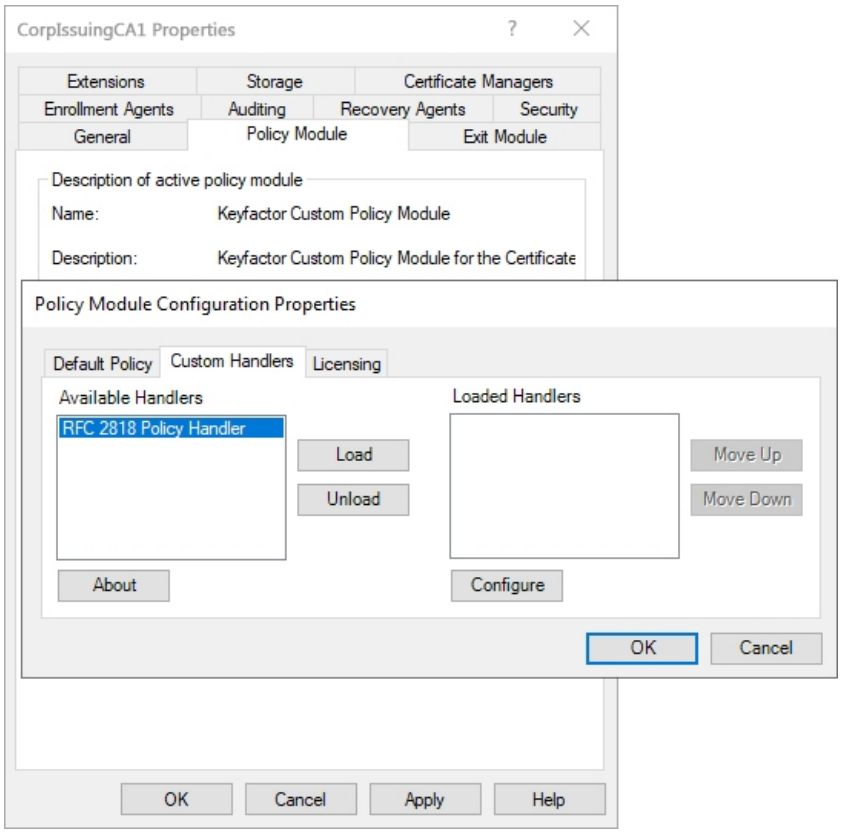


Figure 547: Enable the RFC 2818 Policy Handler

11. On the Custom Handlers tab of the Policy Module Configuration Properties page, highlight the RFC 2818 Policy Handler under Loaded Handlers and click **Configure**.

12. On RFC 2818 Policy Handler configuration dialog, select the templates that should be under management by the RFC 2818 policy handler and click **Add**. Certificate enrollments from any source made using the templates selected here on the configured CA will automatically be assigned a DNS SAN matching the certificate's CN.

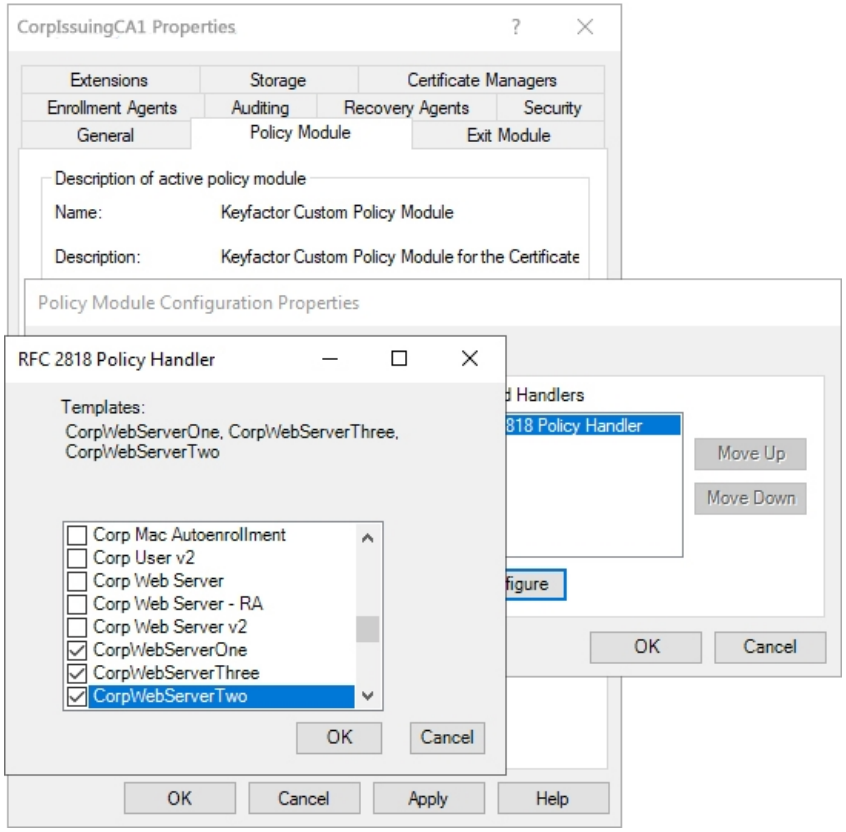


Figure 548: Add Templates for Management with the RFC 2818 Policy Handler

13. Click **OK** as many times as needed to close the configuration dialogs and save the configuration. You will be prompted to restart the CA services.

4.5.3.2 Install the Keyfactor SAN Attribute Policy Handler

To begin the SAN Attribute Policy Handler installation, execute the KeyfactorCAModuleInstaller.msi file from the Keyfactor installation media and install as follows.

1. On the first installation page, click **Next** to begin the setup wizard.

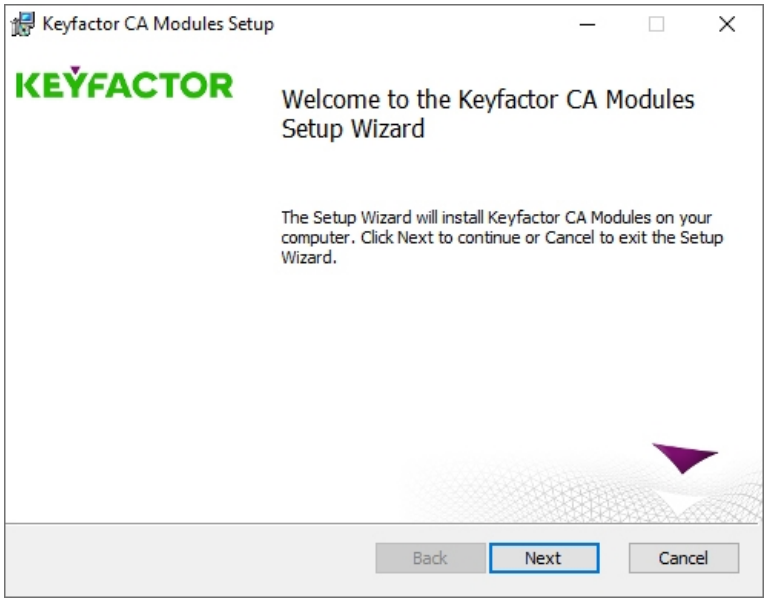


Figure 549: Install SAN Attribute Policy Handler: Begin Setup Wizard

2. On the next page, read and accept the license agreement and click **Next**.

3. On the next page, select the components to install. For the SAN Attribute Policy Handler, deselect all the components except the SAN Attribute Policy Handler component. If desired, you can highlight Keyfactor Custom Policy Module and click **Browse** to select an alternate installation location for the files. The default installation location is:

    C:\Program Files\Keyfactor\Keyfactor CA Modules

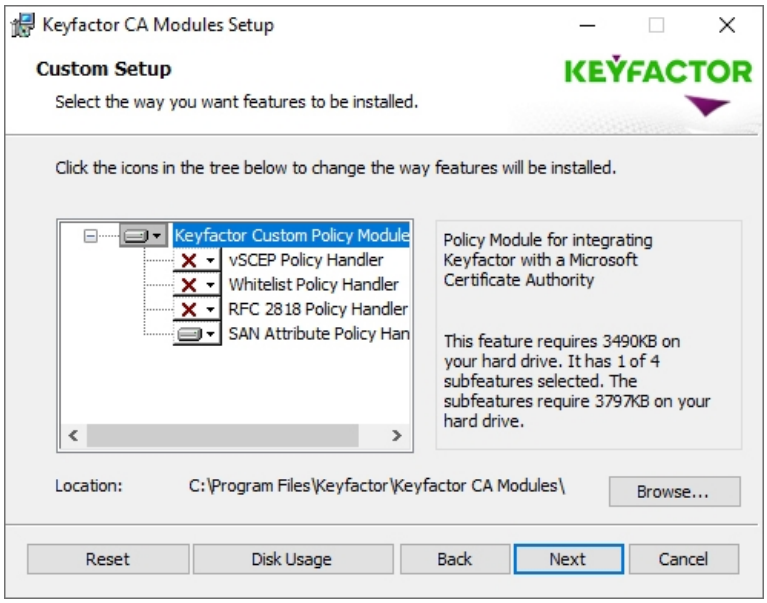


Figure 550: Install SAN Attribute Policy Handler: Select Components

4. On the next screen, click **Install**.

5. On the final installation wizard page, leave the *Launch the CA MMC snap-in now* box selected and click **Finish**. The Microsoft Certification Authority management tool should start automatically. This can take several seconds.

6. In the Certification Authority management tool, right-click the CA name at the top of the tree and choose **Properties**.

7. In the Properties dialog for the CA on the CA Policy Module tab, click **Select**, highlight the **Keyfactor Custom Policy Module** in the Set Active Policy Module dialog and click **OK**.

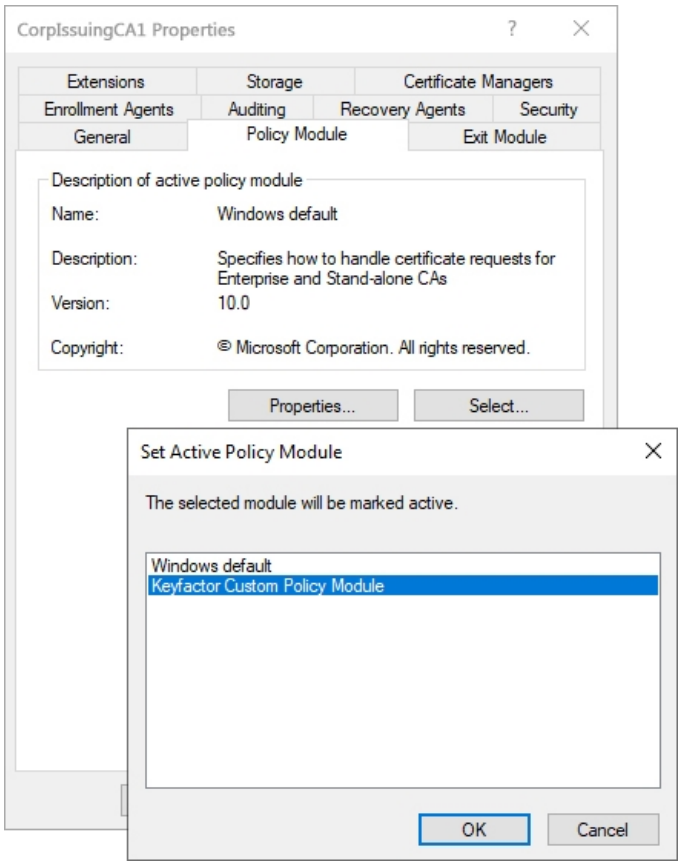


Figure 551: Enable the Keyfactor CA Policy Module

8. In the Properties dialog for the CA on the CA Policy Module tab, click **Properties**.

9. On the Licensing tab of the Policy Module Configuration Properties page, click **Upload License** and browse to locate the license file provided to you by Keyfactor. This file should have the extension CMSLICENSE.

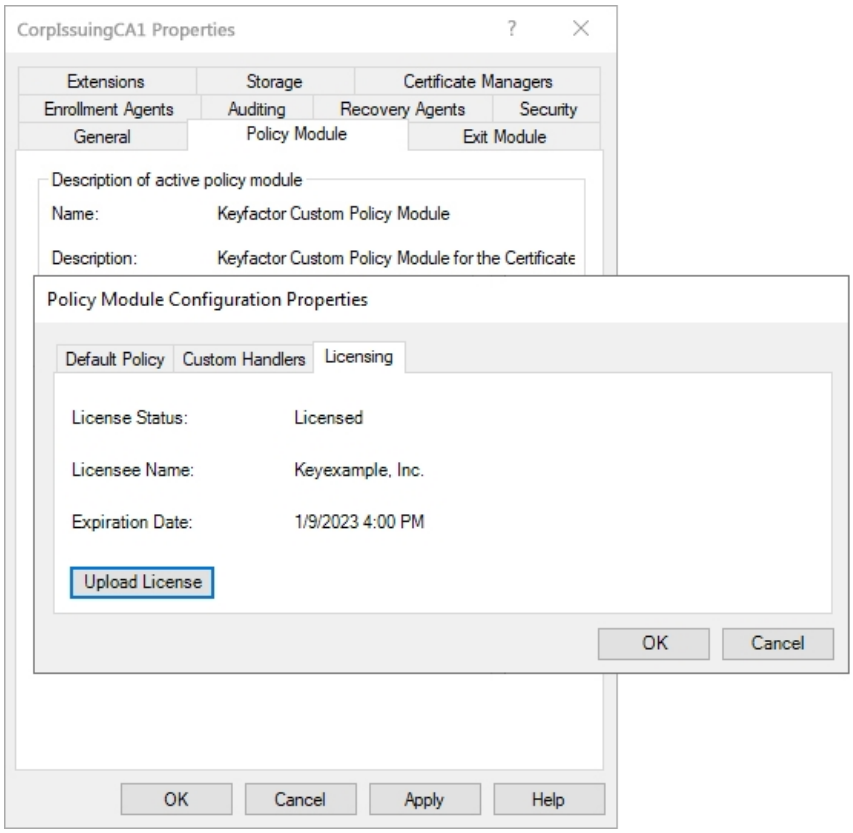


Figure 552: Upload the Keyfactor CA Policy Module License

10. On the Custom Handlers tab of the Policy Module Configuration Properties page, highlight the **SAN Attribute Policy Handler** under Loaded Handlers, click **Load** to move it over to the loaded handlers, and click **OK**.

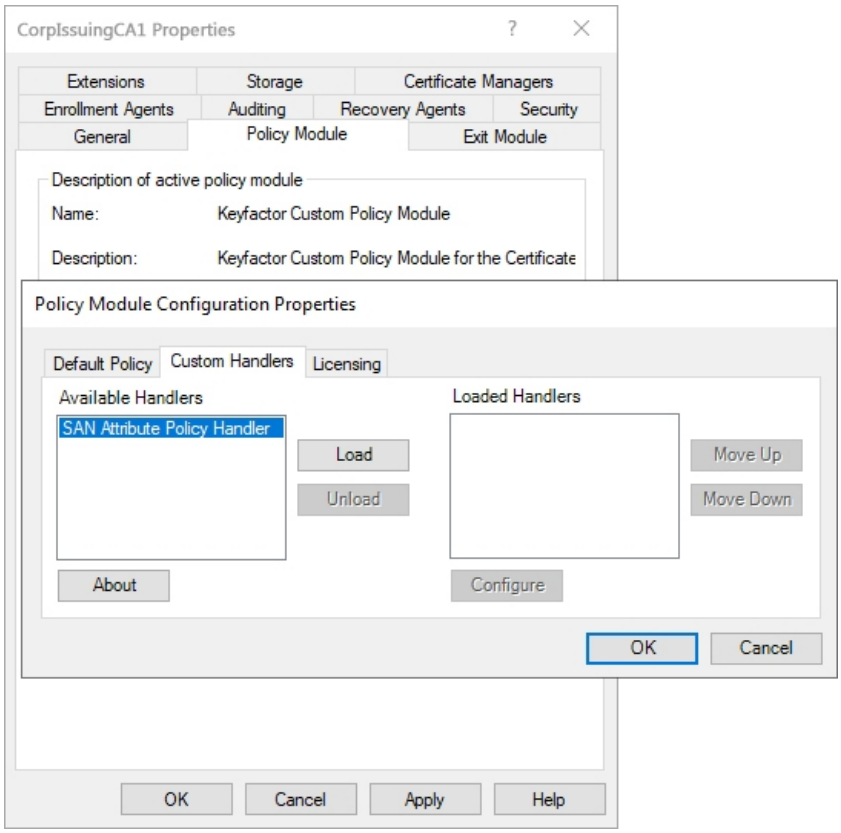


Figure 553: Enable the SAN Attribute Policy Handler

11. On the Custom Handlers tab of the Policy Module Configuration Properties page, highlight the SAN Attribute Policy Handler under Loaded Handlers and click **Configure**.

12. On SAN Attribute Policy Handler configuration dialog, select the templates that should be under management by the SAN Attribute policy handler and click **Add**. Certificate enrollments from any source made using the templates selected here on the configured CA and a CSR enrollment method will allow the addition of SANs not included in the CSR and control the SAN addition functionality on a template-by-template basis without the need to enable the Microsoft CA EDITF_ATTRIBUTESUBJECTALTNAME2 flag.

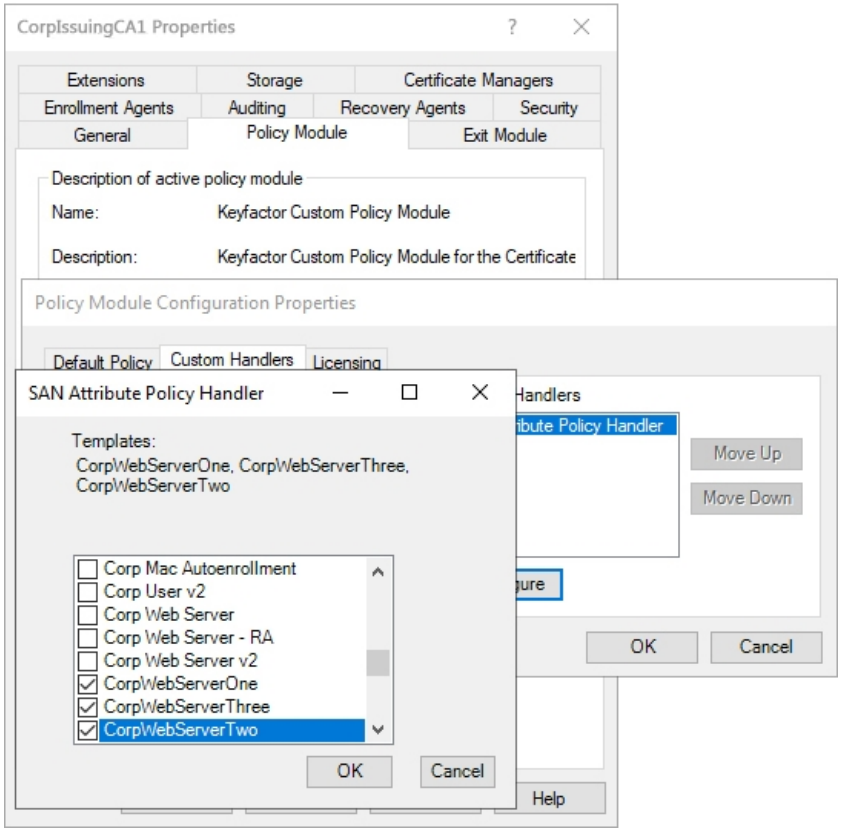


Figure 554: Add Templates for Management with the SAN Attribute Policy Handler

13. Click **OK** as many times as needed to close the configuration dialogs and save the configuration. You will be prompted to restart the CA services.

4.5.3.3 Install the Keyfactor Whitelist Policy Handler

To begin the Whitelist Policy Handler installation, execute the KeyfactorCAModuleInstaller.msi file from the Keyfactor installation media and install as follows.

Note: The following Windows update affects how certificate requests are built when sent to a Microsoft CA and may cause enrollments done outside Keyfactor Command against a Microsoft CA configured with the Whitelist Policy Handler to fail.

https://support.microsoft.com/en-us/topic/kb5014754-certificate-based-authentication-changes-on-windows-domain-controllers-ad2c23b0-15d8-4340-a468-4d4f3b188f16

1. On the first installation page, click **Next** to begin the setup wizard.

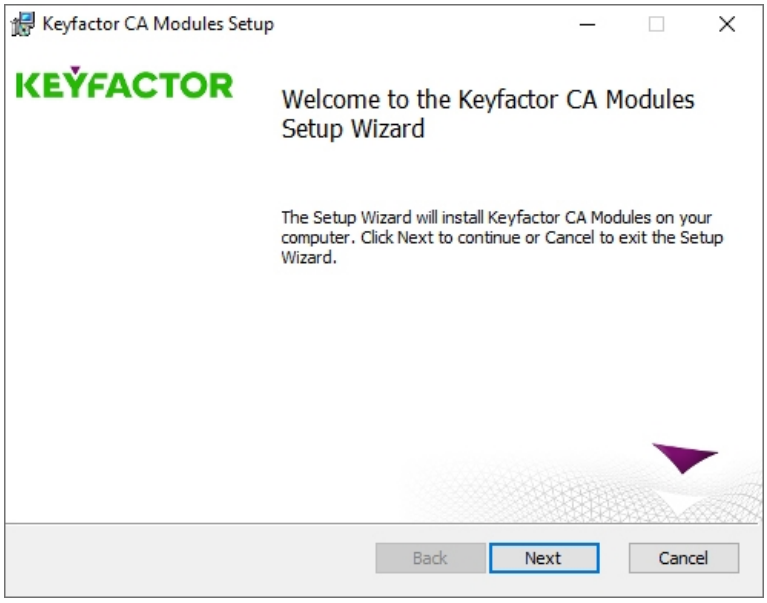


Figure 555: Install Whitelist Policy Handler: Begin Setup Wizard

2. On the next page, read and accept the license agreement and click **Next**.

3. On the next page, select the components to install. For the Whitelist Policy Handler, deselect all the components except the Whitelist Policy Handler component. If desired, you can highlight Keyfactor Custom Policy Module and click **Browse** to select an alternate installation location for the files. The default installation location is:

    C:\Program Files\Keyfactor\Keyfactor CA Modules

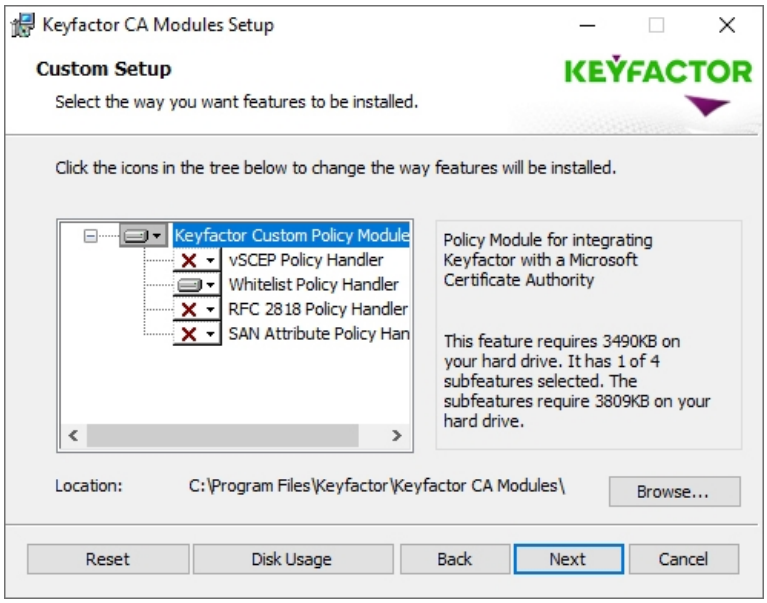


Figure 556: Install Whitelist Policy Handler: Select Components

4. On the next screen, click **Install**.

5. On the final installation wizard page, leave the *Launch the CA MMC snap-in now* box selected and click **Finish**. The Microsoft Certification Authority management tool should start automatically. This can take several seconds.

6. In the Certification Authority management tool, right-click the CA name at the top of the tree and choose **Properties**.

7. In the Properties dialog for the CA on the CA Policy Module tab, click **Select**, highlight the **Keyfactor Custom Policy Module** in the Set Active Policy Module dialog and click **OK**.

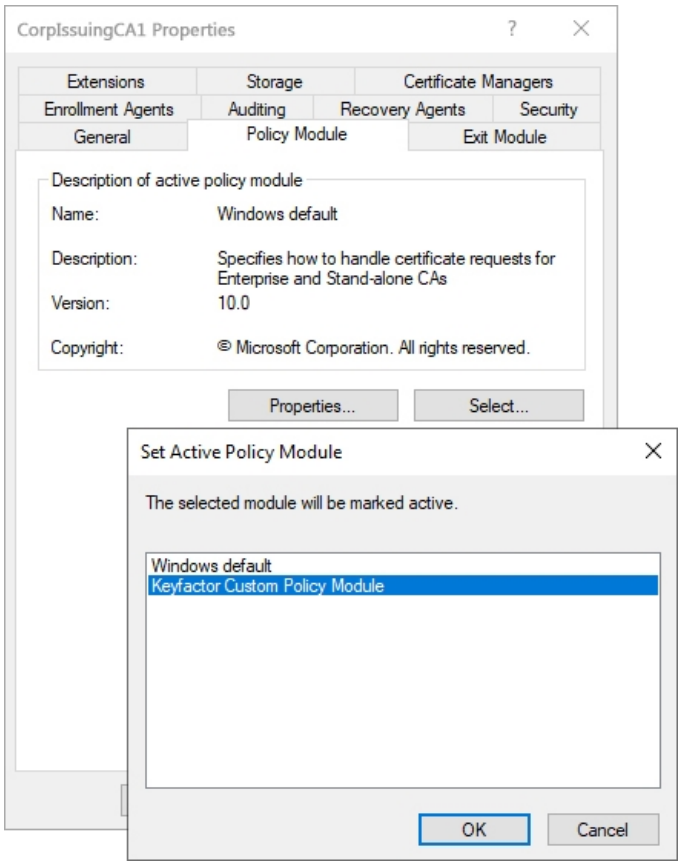


Figure 557: Enable the Keyfactor CA Policy Module

8. In the Properties dialog for the CA on the CA Policy Module tab, click **Properties**.

9. On the Licensing tab of the Policy Module Configuration Properties page, click **Upload License** and browse to locate the license file provided to you by Keyfactor. This file should have the extension CMSLICENSE.

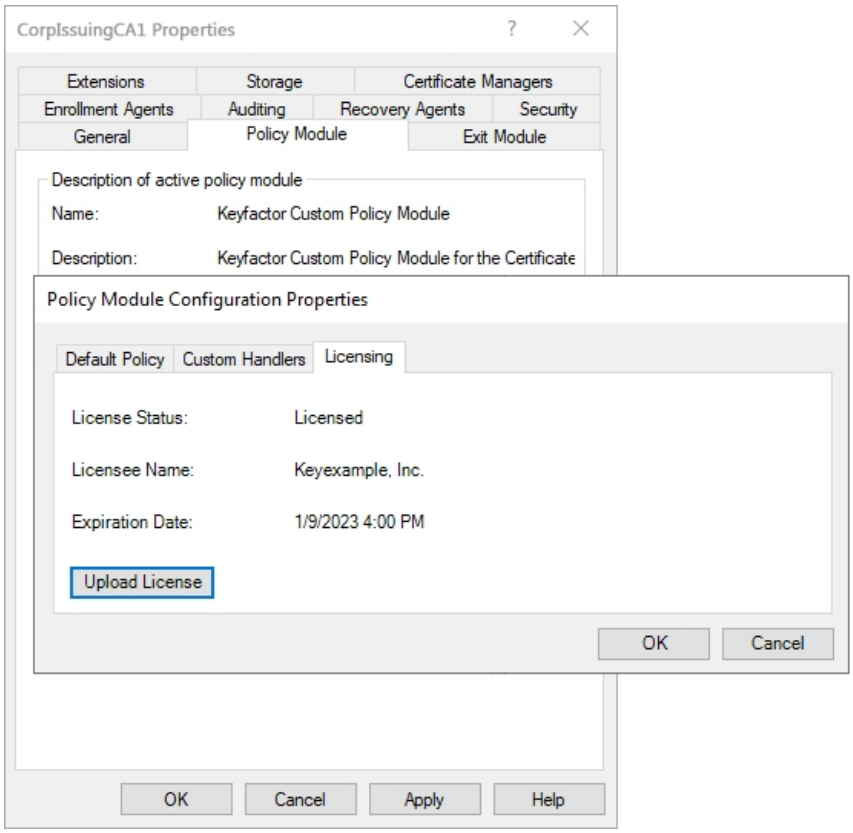


Figure 558: Upload the Keyfactor CA Policy Module License

10. On the Custom Handlers tab of the Policy Module Configuration Properties page, highlight the **CMS Machine Whitelist Policy** on the list of available handlers, click **Load** to move it over to the loaded handlers, and click **OK**.

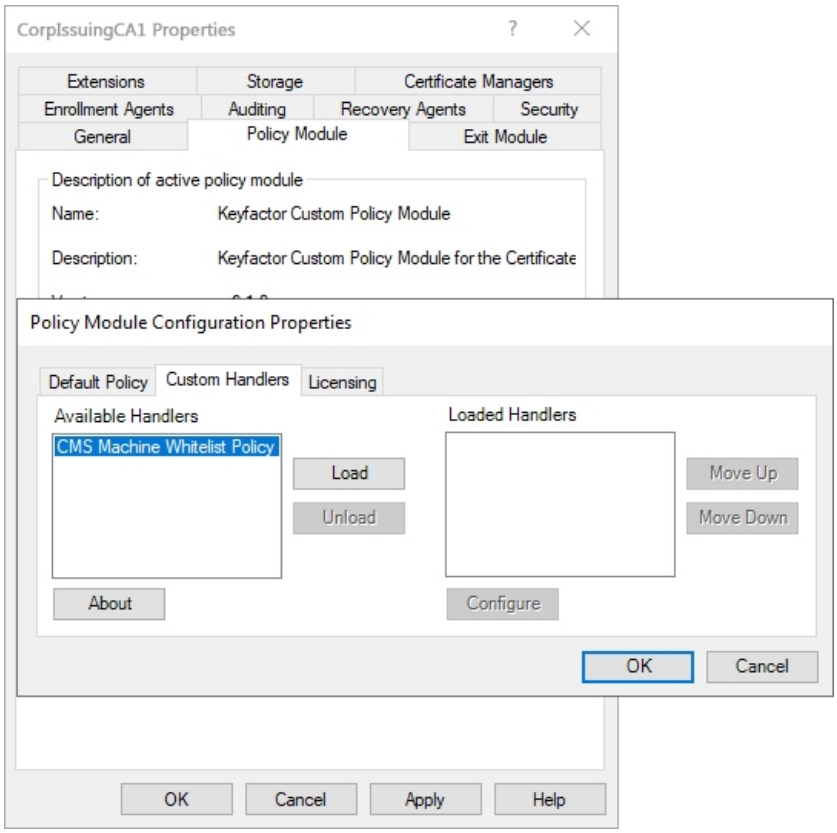


Figure 559: Enable the Whitelist Policy Handler

11. On the Custom Handlers tab of the Policy Module Configuration Properties page, highlight CMS Machine Whitelist Policy under Loaded Handlers and click **Configure**.

12. On the Template tab of the Policy Module Configuration dialog, enter the certificate *template names* (short names), not the template display names, one at a time, of the certificate template(s) you want to manage with the whitelist policy handler and click **Add**. In many cases, the template name is the same as the template display name with the spaces removed. Any templates entered here will be available for enrollment only from machines listed on the Machine Names tab.

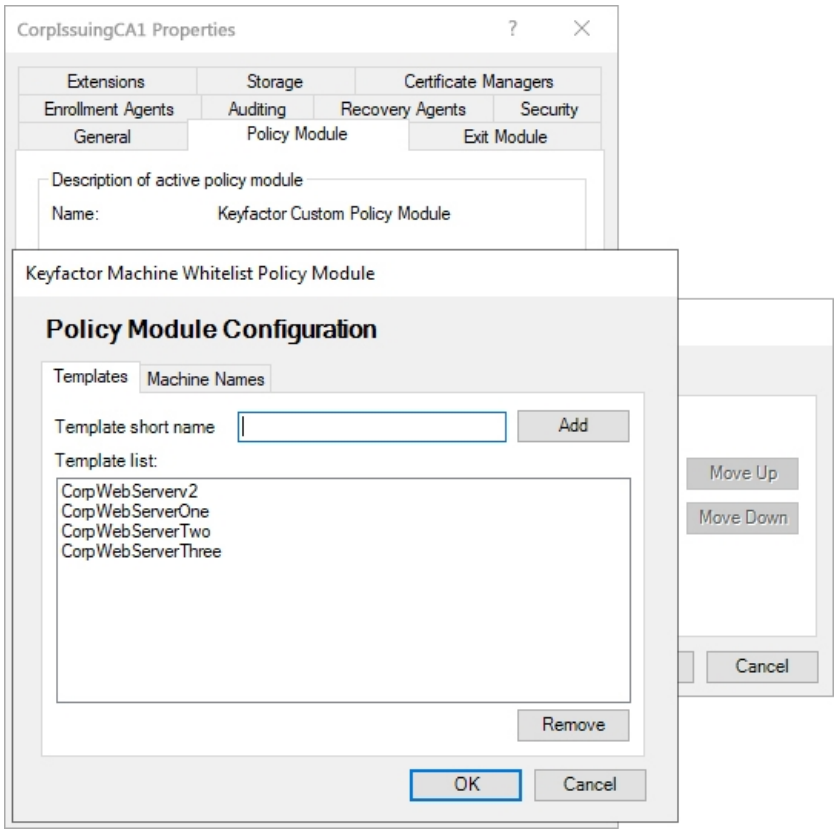


Figure 560: Add Templates for Management with the Whitelist Policy Handler

13. On the Machine Names tab of the Policy Module Configuration dialog, enter the machine names (FQDNs), one at a time, of the machines that you want to manage with the whitelist policy handler and click **Add**. Any machines entered here will be allowed to enroll for the templates listed on the Templates tab.

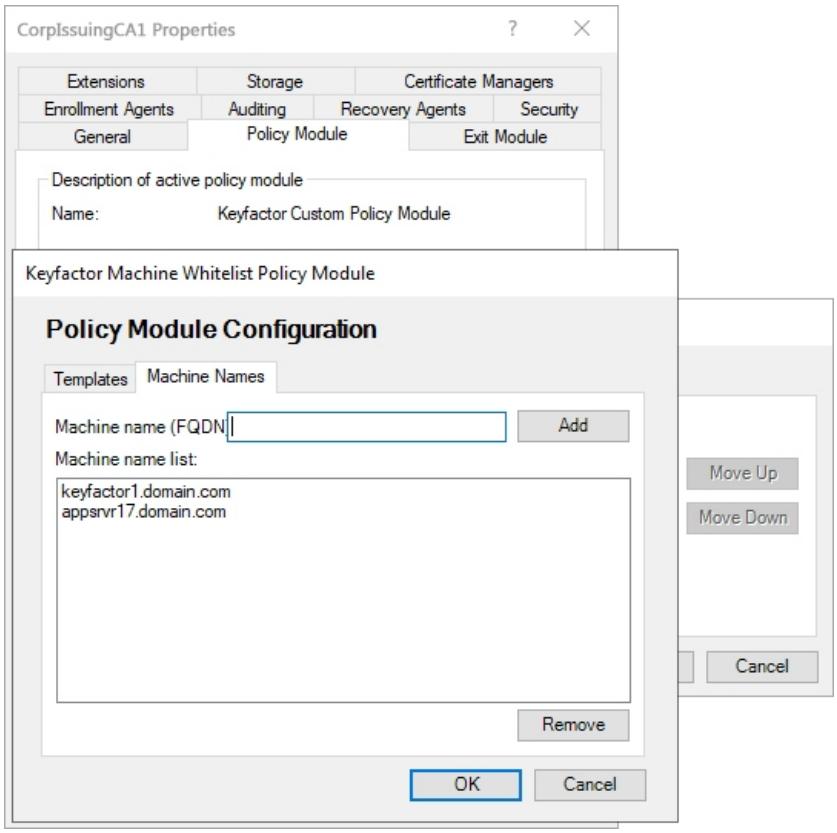


Figure 561: Add Machines for Management with the Whitelist Policy Handler

14. Click **OK** as many times as needed to close the configuration dialogs and save the configuration. You will be prompted to restart the CA services.

4.5.4 Configure Logging for the Keyfactor CA Policy Module

The Keyfactor CA Policy Module provides extensive logging for visibility and troubleshooting. By default, Keyfactor CA Policy Module places its log files in the C:\Keyfactor\logs directory, generates logs at the *Info* logging level and stores the primary logs for two days before deleting them.

To configure logging:

1. On the policy module server where you wish to adjust logging, open a text editor (e.g. Notepad) using the “Run as administrator” option.

2. In the text editor, browse to open the Nlog.config file for the Keyfactor CA Policy Module The file is located in the installation directory for the product, which is the following by default:

    C:\Program Files\Keyfactor\Keyfactor CA Modules\NLog.config

3. Your Nlog.config file may have a slightly different layout than shown here, but it will contain the five fields highlighted in Figure 562: Keyfactor CA Policy Module NLog.config File. The fields you may wish to edit are:

    - `fileName="C:\Keyfactor\Logs\Keyfactor_CA_Log.txt"`

      The path and file name of the active policy module log file, referencing the logDirectory variable.

      > **Important:** If you choose to change the path for storage of the log files, you will need to create the new directory (e.g. D:\KeyfactorLogs) and grant the service account under which the Active Directory Certificate Services service is running full control permissions on this directory.

    - `archiveFileName="C:\Keyfactor\Logs\Keyfactor_CA_Log_Archive_{#}.txt"`

      The path and file name of previous days' orchestrator log files, referencing the logDirectory variable. The orchestrator rotates log files daily and names the previous files using this naming convention.

    - `maxArchiveFiles="2"`

      The number of archive files to retain before deletion.

    - `name="*" minlevel="Info" writeTo="logfile"`

      The level of log detail that should be generated and output to the log file. The default *Info* level logs error and some informational data but at a minimal level to avoid generating large log files. For troubleshooting, it may be desirable to set the logging level to *Debug* or *Trace*. Available log levels (in order of increasing verbosity) are:

        - OFF—No logging
        - FATAL—Log severe errors that cause early termination
        - ERROR—Log severe errors and other runtime errors or unexpected conditions that may not cause early termination
        - WARN—Log errors and use of deprecated APIs, poor use of APIs, *almost* errors, and other runtime situations that are undesirable or unexpected but not necessarily *wrong*
        - INFO—Log all of the above plus runtime events (startup/shutdown)
        - DEBUG—Log all of the above plus detailed information on the flow through the system
        - TRACE—Maximum log information—this option can generate VERY large log files

```
<targets>
  <target name="buffered_wrapper" xsi:type="BufferingWrapper" slidingTimeout="true" bufferSize="500" flushTimeout="500">
    <target xsi:type="File" name="logfile" fileName="C:\Keyfactor\logs\Keyfactor_CA_Log.txt" layout="${longdate} ${logger} [${level}] - ${message}"
            archiveFileName="C:\Keyfactor\logs\Keyfactor_CA_Log_Archive_{#}.txt" archiveEvery="Day" archiveNumbering="Rolling" maxArchiveFiles="2"/>
  </target>
  <target xsi:type="OutputDebugString" name="String" layout="${longdate} ${logger}::${message}"/>
  <target xsi:type="Debugger" name="debugger" layout="${longdate} ${logger}::${message}"/>
  <target xsi:type="Console" name="console" layout="${logger} ${message}"/>
  <target xsi:type="EventLog" name="eventLog" source="Keyfactor CA Modules"
          eventId="${event-properties:item=eventID}" category="${event-properties:item=categoryID}" layout="${event-properties:item=message}" />
</targets>
<rules>
  <!-- Don't write events to the log file (log file should contain different, more verbose, logging) -->
  <logger name="*-EVENT" minlevel="Info" writeTo="eventLog" final="true" />
  <logger name="*" minlevel="Info"  writeTo="logfile" />
</rules>
```

Figure 562: Keyfactor CA Policy Module NLog.config File

4.5.5 Add Non-Keyfactor SCEP Servers to the Ignore List

This step only needs to be completed if you installed the Keyfactor CA Policy Module with the vSCEP™ Policy Handler.

If your CA for that issues certificates based on SCEP challenges is used by multiple SCEP servers, you will need to add SCEP servers not used for Keyfactor Command vSCEP API requests to the ignore list on the CA running the vSCEP™ Policy Handler. This will allow the vSCEP™ Policy Handler to ignore requests (passing them through to the CA) from the listed SCEP servers. Without this feature, SCEP challenges from non-Keyfactor Command servers would be denied because no data exists against which to verify the certificate details.

The vSCEP™ Policy Handler reads the ignore list in the registry string value **CCMBlacklist**. This field contains a semicolon-delimited list of SCEP server host names from which all certificate requests should be ignored by the Keyfactor CA Policy Module and passed through to the CA. The **CCMBlacklist** setting can be found in the following registry location:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Certified Security Solutions\vSCEP\Configuration
```

The **CCMBlacklist** registry value does not exist by default. Use the Registry Editor (regedit) to create the **CCMBLacklist** value as a DWORD and populate it with the SCEP server FQDNs for any SCEP servers whose requests should bypass the vSCEP™ Policy Handler.

4.6 Appendices

- Appendix - Troubleshooting Logi Log Files below
- Appendix - Logi Load Balancing: Keyfactor Command Configuration Wizard Setup on page 2872
- Appendix - Configuration Wizard Errors in the Logs on page 2874

4.6.1 Appendix - Troubleshooting Logi Log Files

When troubleshooting Logi, the first thing to try is setting the *Debug Embedded Reports* application setting to **True** (see Application Settings: Console Tab on page 602 in the *Keyfactor Command Reference Guide*). This allows the reports to output errors with debug level information if they generate

errors. If this does not generate the information necessary to resolve the problem, it can sometimes be helpful to modify the Keyfactor Analysis web.config file to allow IIS to show the actual error the application is experiencing at a lower level. To configure this:

1. Browse to the *Logi* directory under the installed directory for your Keyfactor Command implementation. By default, this is:

   C:\Program Files\Keyfactor\Keyfactor Platform\Logi

2. Using a text editor opened with the "Run as administrator" option, open the web.config file for editing.

3. Find the <customErrors mode="RemoteOnly"/> section and change this to <customErrors mode=e="On"/>.

4. Look for the debug output in the *Logi\rdDownload* directory under the installed directory for your Keyfactor Command implementation. By default, this is:

   C:\Program Files\Keyfactor\Keyfactor Platform\Logi\rdDownload

Tip: If you do not find debug output when running a report manually in the Management Portal, try scheduling a report for delivery via email or saving to disk using the Report Manager. Debugging operates differently for these two modes of running a report.

Note: When the portal experiences a 500 error, Logi logs will not be written to the usual output directory.

```
<system.web>
  <!--  DYNAMIC DEBUG COMPILATION
        Set compilation debug="true" to insert debugging symbols (.pdb information)
        into the compiled page. Because this creates a larger file that executes
        more slowly, you should set this value to true only when debugging and to
        false at all other times. For more information, refer to the documentation about
        debugging ASP.NET files.
  -->
  <compilation defaultLanguage="vb" debug="true" />

  <!--  CUSTOM ERROR MESSAGES
        Set customErrors mode="On" or "RemoteOnly" to enable custom error messages, "Off" to disable.
        Add <error> tags for each of the errors you want to handle.
  -->
  <customErrors mode="RemoteOnly" />
  <!--  AUTHENTICATION
        This section sets the authentication policies of the application. Possible modes are "Windows",
        "Forms", "Passport" and "None"
  -->

  <authentication mode="Windows" />
```

Figure 563: Logi web.config

4.6.2 Appendix - Logi Load Balancing: Keyfactor Command Configuration Wizard Setup

In order for the Keyfactor Command Management Portal Dashboard and Reports to load when using a load balancer, the Keyfactor Command Configuration Wizard should have the following configuration on each of the application servers:

- On the Keyfactor Command Portal Tab, the Host Name must be the Load Balanced URL.

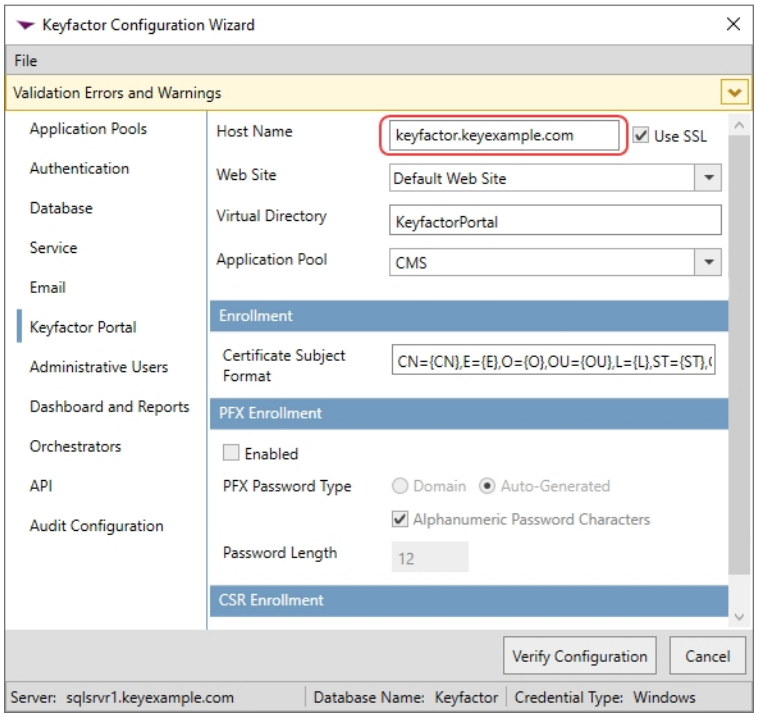


Figure 564: Logi Configuration Settings—Keyfactor Command Portal Tab

- Dashboard and Reports Tab:
  - The Host Name must be the Load Balanced URL. This is the host name that the Management Portal server uses to connect to the Logi Analytics Platform, and it therefore needs to be the name used on the internal side of the network.

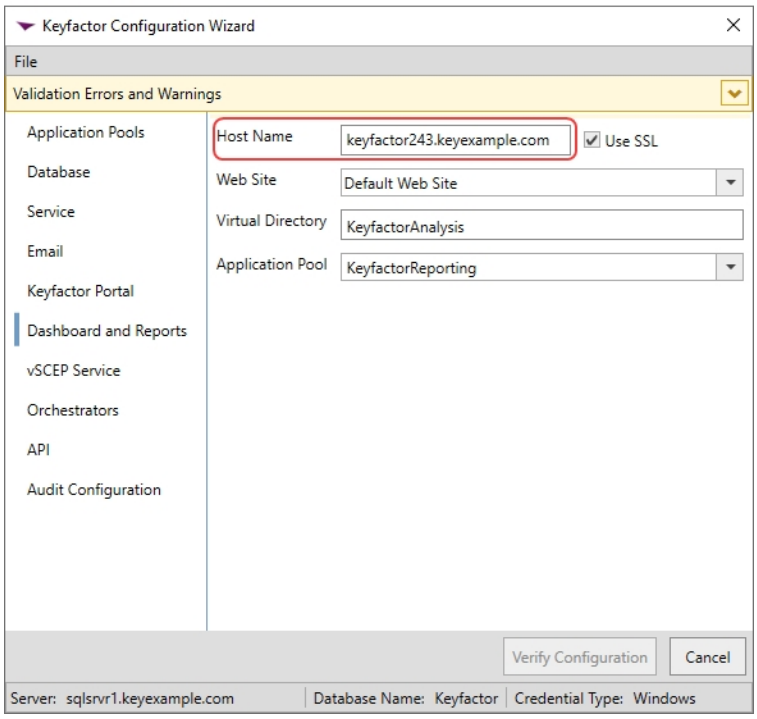


Figure 565: Logi Configuration Settings—Keyfactor Command Dashboards and Reports Tab

Tip: All IP Addresses that could be used internally to connect to the Logi application must be in the Dashboard and Reports configuration section in the configuration wizard. This includes the application host IPs and the load balancer IPs. It is also recommended that the host file is modified to map the load balancer URL to the local IP address.

- Load Balancer

  On the load balancer, a new rewrite rule needs to be made that changes the outbound URL from the application servers. Logi sends the HostName.domain.com/KeyfactorAnalysis URL back to the browser instead of the LoadBalancer.URL.com/KeyfactorAnalysis URL that the browser needs to complete the Logi authorization. In short, an Outbound rewrite rule needs to be created on the Load Balancer that does the following: HostName_URL/KeyfactorAnalysis needs to be converted to LoadBalancer_URL/keyfactorAnalysis

- Load Balancer - Session Affinity

  There are two load balancing scenarios based on user session management: Sticky session (recommended) and Non-Sticky session. In the Sticky session scenario, each user is assigned to a server by the load balancer and all the requests sent by this user are answered by the same server, for as long as the user's session persist. This is the recommended approach and does not require you to centralize the rdDataCache folder of the application. We strongly recommend using Sticky session instead. You can learn more about Load Balancing with Info applications on:

  https://devnet.logianalytics.com/hc/en-us/articles/4419707733783-Load-Balancing-Configuration

4.6.3 Appendix - Configuration Wizard Errors in the Logs

If an incoming web request runs before the configuration is fully completed, you may encounter the following errors in the Management Portal log file after upgrading. These errors are not something to be worried about. They just indicate that the web request was still looking at an old version of the database prior to it being completely upgraded.

```
2021-11-15 12:41:36.5155 Keyfactor.EF.KeyfactorExecutionStrategy [Error] - SqlException
with number 207 occurred, not attempting to retry the connection.
2021-11-15 12:41:36.5780 Keyfactor.EF.KeyfactorExecutionStrategy [Error] - Invalid
column name 'Immutable'.

Invalid column name 'SubscriberTerms'.

2021-11-15 12:41:36.6249 ASP.global_asax [Error] - An uncaught application error
occurred: An error occurred while executing the command definition. See the inner
exception for details.

2021-11-15 12:41:37.3281 ASP.global_asax [Error] - An uncaught application error
occurred: An item with the same key has already been added.

at System.ThrowHelper.ThrowArgumentException(ExceptionResource resource)
```

5.0 Installing Orchestrators

Keyfactor offers several orchestrators (a.k.a. agents) that may be used to interact with and enhance the functionality of the Keyfactor Command Server.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

This guide covers installation of the following orchestrators:

- Keyfactor Universal Orchestrator
  The Keyfactor Universal Orchestrator runs on Windows Servers, Linux servers, and in Linux containers. It can be used to:
  - Run SSL discovery and monitoring tasks.
  - Manage synchronization of certificate authorities in remote forests (installations on Windows only).
  - Collect logs from the orchestrator for central review (full server installations only).
  - Run custom jobs to provide certificate management capabilities on a variety of platforms and devices.
  - Run custom jobs to execute tasks outside the standard list of certificate management functions. This powerful feature can execute just about any job that requires processing on the orchestrator and submitting data back to Keyfactor Command.

  As of this release, the following functions, some of which were part of the Keyfactor Windows Orchestrator, are now included among the custom extensions supported for the Keyfactor Universal Orchestrator:
  - Interact with Amazon Web Services (AWS) resources for certificate management.
  - Interact with Citrix NetScaler devices for certificate management.
  - Interact with F5 devices for certificate management.
  - Interact with Windows servers (a.k.a. IIS certificate stores), create new bindings for IIS web sites and manage certificates in both the Web Hosting certificate store and the Personal certificate store.
  - Remote Java keystore certificate management.
  - Remote PEM store certificate management.
  - Remote PKCS12 store certificate management.

  These custom extensions and more are publicly available at:

  https://keyfactor.github.io/integrations-catalog/content/orchestrator

The final release of the Keyfactor Windows Orchestrator was version 8.7. This version of the Keyfactor Windows Orchestrator is not compatible with Keyfactor Command version 11.0. Customers should migrate to the Keyfactor Universal Orchestrator with custom extensions as needed.

- Keyfactor Bash Orchestrator
  The Keyfactor Bash Orchestrator runs on Linux servers and is used to perform discovery and management of SSH public keys, including installation of new keys and automated removal of unauthorized keys.
- Keyfactor Java Agent
  The Keyfactor Java Agent runs on Windows or Linux servers and is used to perform discovery of Java keystores and PEM certificate stores, to inventory discovered stores, and to push certificates out to stores as needed. In addition, the Keyfactor Java Agent can be extended to create custom certificate store jobs.

  Important: The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

  For more information, see Installing Custom-Built Extensions on page 2940.

Keyfactor also offers a variety of tools to allow users to develop custom orchestrators and extensions, including:

- Keyfactor AnyAgent Framework

  The AnyAgent capability of the Keyfactor Universal Orchestrator and Java Agent allows management of certificates regardless of source or location by allowing customers to implement custom agent functionality.
- Keyfactor Integration SDK

  The Keyfactor Integration SDK (software development kit) includes a variety of tools for building a custom orchestrator, including the Keyfactor Native Agent, which is a reference implementation intended for customers wanting to include Keyfactor Command certificate store management functionality in embedded or other platforms.
- Keyfactor Orchestrator NuGet Package

  The Keyfactor Orchestrator NuGet package is designed to allow customers to build custom extensions for the Keyfactor Universal Orchestrator.
- Keyfactor GitHub Site

  Keyfactor offers several publicly available integrations and plugins for the Keyfactor platform in the Keyfactor GitHub. Find all the latest developer tools and resources to integrate the Keyfactor platform with your PKI, Cloud, and DevOps infrastructure.

  https://keyfactor.github.io/

These tools for developing custom orchestrators and extensions are not documented in this guide. For more information about these and other custom orchestrator solutions, contact your Keyfactor representative.

> ⚠ **Important:** The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:
>
> https://github.com/Keyfactor/remote-file-orchestrator
>
> For more information, see Installing Custom-Built Extensions on page 2940.

5.1 Orchestrator Job Overview

Keyfactor orchestrators can be used to perform a wide variety of jobs. Out of the box, orchestrators can manage certificate stores, manage SSH keys, perform SSL scanning, fetch system logs, and synchronize certificates from CAs in remote forests. Orchestrator jobs fall into these broad types:

- Certificate Store Jobs

  This type of job includes the built-in jobs for managing certificate stores, based on the type(s) of certificate stores supported by the orchestrator, and custom-built certificate store jobs that can be added with an extension (see Installing Custom-Built Extensions on page 2940) or script (see Configuring Script-Based Certificate Store Jobs on page 2946).

  Certificate store jobs (built-in or custom-built), are managed in Keyfactor Command with certificate store types. If you're adding a custom-built certificate store job, you'll need to add a user-defined certificate store type to go with it (see Certificate Store Types on page 700 and Certificate Store Types on page 1531).

- Custom Jobs

  This type of job is intended to implement just about anything else you need an orchestrator to do other than manage certificate stores. The built-in fetch logs job is an example of a custom job.

  Custom jobs are managed in Keyfactor Command with custom job types. If you're adding a custom job, you'll need to add a custom job type to go with it (see Custom Job Types on page 1594).

  Custom jobs are supported only by the Keyfactor Universal Orchestrator.

- Other Jobs

  This type of job includes the built-in jobs for SSL scanning and certificate synchronization from remote CAs.

Prescripts and Postscripts

All of the job types supported by the Keyfactor Universal Orchestrator—including the built-in jobs—support executing a prescript and/or postscript as part of the job. A prescript might be used to fetch credentials from a privilege access management (PAM) solution to pass in to the username and password parameters for a certificate store. A postscript might be used to restart the web service (e.g.

Apache) after performing a management job to replace the certificate in the certificate store for the web server. Prescripts and postscripts for all types of jobs are configured similarly to the description provided for installing custom-built extensions (see Installing Custom-Built Extensions on page 2940).

Note: The prescript and postscript functionality of the Keyfactor Universal Orchestrator has been replaced by other functionality in Keyfactor Command such as that provided by Keyfactor Command workflows (see Workflow Definitions on page 230). As a result, prescript and postscript functionality has been deprecated and will be removed from a future release.

Orchestrator Job Flow

An orchestrator job begins when an orchestrator queries Keyfactor Command to ask for jobs and the Keyfactor Command orchestrator API returns a list of the jobs the orchestrator needs to run. The flow continues as shown in the following chart.

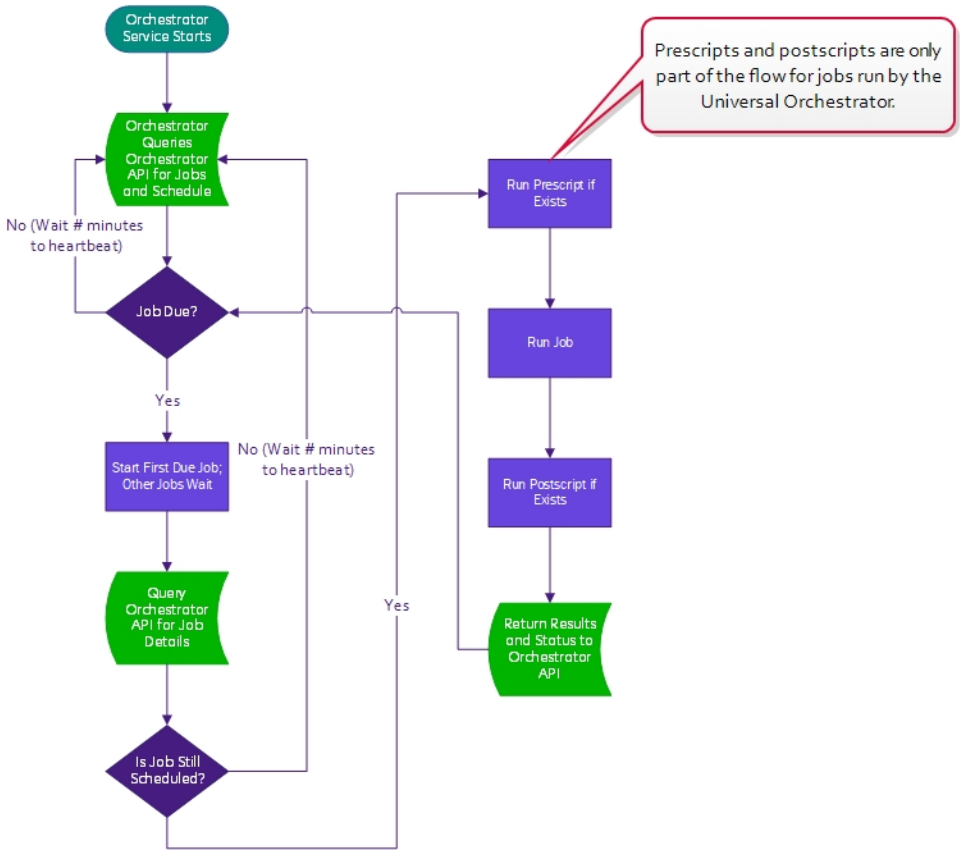


Figure 566: Orchestrator Job Flow

5.2 Universal Orchestrator

The Keyfactor Universal Orchestrator is designed to run jobs at the request of the Keyfactor Command server. Jobs primarily perform certificate management tasks, but other types of operations are also supported. The orchestrator operates as a .NET Core based service on a Windows server, Linux server, or in a Linux container and communicates with a Keyfactor Command server to receive job tasks and report job results. Along with the job results, data can be returned to the Keyfactor Command server and stored in the Keyfactor Command SQL database. Extensions are hosted by the orchestrator and implement the jobs to be executed.

The orchestrator includes these built-in extensions:

- Discover and monitor certificates at TLS 1.3 endpoints either within the local network or across the internet using any of the 5 ciphersuites mentioned in appendix B.4 of RFC 8446. Certificates from the results of SSL discovery and monitoring are imported into Keyfactor Command for viewing, reporting and alerting purposes. Scanning using server name indication (SNI) is supported.
- Retrieve logs generated on the orchestrator via the Keyfactor Command Management Portal. This task returns up to 2 MB of log data from the end of the orchestrator log file to be viewed in the Management Portal. This features is supported only on full server installations.
- Manage certificates from remote Microsoft Certificate Authorities (CAs) using the Management Portal. Certificates from remote CAs can be imported into Keyfactor Command for viewing, reporting and alerting purposes. This feature is supported only on Windows installations.

  If the remote CA is domain-joined to a domain in the remote forest, the Universal Orchestrator may be installed on the CA itself or on a separate server joined to a domain in the same forest (generally a server in the same domain as the CA). Multiple CAs in the same remote forest can be managed with a single Universal Orchestrator server. However, if the remote CA is not domain-joined, the Universal Orchestrator must be installed on the remote CA server.

  Note: The Universal Orchestrator does not support certificate enrollment for remote CAs. If you need this capability, you will need to use the *Explicit Credentials* option in the Management Portal CA configuration (see Adding or Modifying a CA Record on page 354).

In addition, two types of custom extensions are supported:

- Manage and deliver certificates to certificate stores on various platforms and devices using custom certificate store types and orchestrator jobs in the Keyfactor Command Management Portal. Custom extensions may be developed by Keyfactor or end users. Keyfactor offers several publicly available custom extensions for the Universal Orchestrator in the Keyfactor GitHub:

  https://keyfactor.github.io/integrations-catalog/content/orchestrator

With the custom extensions available from the Keyfactor GitHub, you can manage Windows certificate stores (IIS), JKS stores, PEM stores, F5 devices, Citrix NetScaler devices, AWS resources and more (see Installing Custom-Built Extensions on page 2940).

For more information about custom extensions, contact your Keyfactor representative.

- Run custom jobs on the orchestrator that fall outside the standard certificate management tasks. With custom jobs, you can perform operations locally on the orchestrator—or initiate them remotely across the network—and then report results back to Keyfactor Command along with data collected from the jobs, if any.

5.2.1 Preparing for the Universal Orchestrator

This section describes the steps that need to be taken prior to a Keyfactor Universal Orchestrator installation to install the prerequisites, create the required supporting components, and gather the necessary information to complete the Universal Orchestrator installation and configuration process.

5.2.1.1 System Requirements

The Keyfactor Universal Orchestrator is supported on the following operating systems:

- Windows Server 2019 or Windows Server 2022
- Oracle Linux 7 or higher
- Red Hat Enterprise 7 or higher
- Ubuntu 16 or higher

Note: Older versions of the Universal Orchestrator will work with newer versions of Keyfactor Command, but not the other way around (see the Compatibility Matrix). The current version of the Universal Orchestrator requires Keyfactor Command version 10.0 or greater.

Important: Microsoft support for .NET Runtime version 3.1 was deprecated at the end of 2022. Instructions for upgrading to version 6.0 are included in the tip, below.

Windows Server Application Requirements

The Universal Orchestrator has the following requirements on Windows.

- The orchestrator requires the Microsoft .NET Runtime version 6.0 (x64). Version 6.0 is available for download from Microsoft:

  https://dotnet.microsoft.com/download/dotnet/6.0/runtime

  You need only the .NET Runtime (x64), not the ASP.NET Core Runtime or ASP.NET Core Hosting Bundle. At the above link, this would be the **Download x64** option under the *Run console apps* heading.

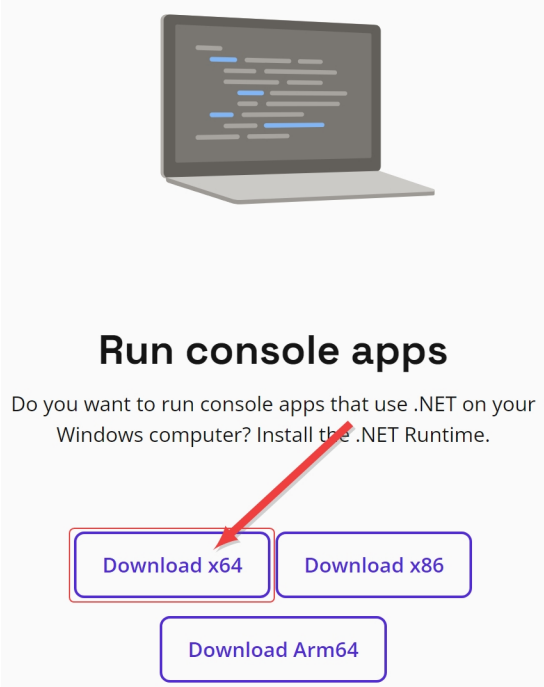


Figure 567: Select the Download x64 Option Under Run Console Apps

You can use the following PowerShell command to check the .NET core version(s) installed on a server (if any):

```
dotnet --list-runtimes
```

Output from this command will look something like this if you have the correct 6.0 x64 version of the .NET Runtime installed (notice the path is in C:\Program Files, not C:\Program Files (x86), indicating this is the x64 version):

```
Microsoft.NETCore.App 6.0.11 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
```

- If you intend to use the orchestrator to manage certificates in remote Windows machine certificate stores (servers other than the server on which the orchestrator is installed) using the IIS Certificate Store Manager extension or Java Keystores, PKCS12 files, PEM files, DER files, or IBM Key Database files on Windows servers with the Remote File Certificate Store Management extension (see Installing Custom-Built Extensions on page 2940), make sure that TCP port 5985 or 5986 is open between the orchestrator and the remote servers (see Configure Windows Targets for Remote Management on page 2935).
- If you intend to use the orchestrator to manage certificates from remote Microsoft CAs (CAs outside the forest in which Keyfactor Command is installed or forests in a two-way trust with this forest), the orchestrator requires the Microsoft Visual C++ 2019 (or later) redistributable for x64. This is available for download from Microsoft:

  https://aka.ms/vs/16/release/vc_redist.x64.exe

The Microsoft Visual C++ Redistributable appears as an application in the Windows Apps & features.

Linux Server Application Requirements

The following applications are required in order to install the Universal Orchestrator on Linux servers.

Microsoft .NET 6.0 Runtime

The orchestrator requires the Microsoft .NET Runtime version 6.0 (x64). Information about this is available from Microsoft:

https://docs.microsoft.com/en-us/dotnet/core/install/linux

You need only the .NET Runtime (x64), not the ASP.NET Core Runtime, but it won't hurt anything to install both the .NET and ASP.NET Core runtimes as suggested in the Microsoft documentation for installing .NET on Linux.

For the most part, it can be installed via the OS package manager. The method to complete this varies depending on the Linux operating system. For example, for Ubuntu 20.04, the following commands will install the correct version of .NET:

```
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb
 sudo dpkg -i packages-microsoft-prod.deb
 sudo apt-get update
 sudo apt-get install apt-transport-https
 sudo apt-get install dotnet-runtime-6.0
```

You can use the following command to check the .NET version installed on a server (if any):

```
dotnet --list-runtimes
```

Output from this command will look something like this if you have the correct 6.0 version of the .NET Runtime installed:

```
Microsoft.NETCore.App 6.0.6 [/usr/share/dotnet/shared/Microsoft.NETCore.App]
```

jq

The orchestrator can only be installed on a Linux server that has jq installed. You can use the following command to check the jq version of a server:

```
jq --version
```

systemd

The orchestrator requires a Linux server that uses the systemd service manager. You can use the following command to test whether a system is running systemd:

```
ps -p 1
```

bash

The orchestrator can only be installed on a Linux server that is running bash version 4.3 or higher. You can use the following command to check the bash version of a server:

```
bash --version
```

curl

The orchestrator can only be installed on a Linux server that has curl installed. You can use the following command to check the curl version of a server:

```
curl --version
```

Linux Container Application Requirements

The following applications are required in order to install the Universal Orchestrator in Linux containers.

Containerization Solution

The orchestrator needs a containerization solution in which to run. Keyfactor has tested with Docker and Kubernetes.

Tip: If you have an existing installation of the Universal Orchestrator using the older Microsoft .NET Runtime version 3.1, you do not need to reinstall the orchestrator to upgrade the .NET version.
To update your existing Universal Orchestrator to the latest .NET version:

1. On the Universal Orchestrator machine, browse to locate the Orchestrator.runtimeconfig.json file in your installation directory. By default, this is:

   ```
   Windows: C:\Program Files\Keyfactor\Keyfactor
   Orchestrator\Orchestrator.runtimeconfig.json
   Linux: /opt/keyfactor/orchestrator/Orchestrator.runtimeconfig.json
   ```

2. Using a text editor, open the Orchestrator.runtimeconfig.json file for editing and add the following property to the runtimeOptions section:

   ```
   "rollForward": "LatestMajor"
   ```

Being sure to add a comma at the end of the previous row, resulting in a final file that looks something like:

```
{
   "runtimeOptions": {
      "tfm": "netcoreapp3.1",
      "framework": {
         "name": "Microsoft.NETCore.App",
         "version": "3.1.0"
      },
      "rollForward": "LatestMajor"
   }
}
```

3. Save the Orchestrator.runtimeconfig.json file.
4. Uninstall the Microsoft .NET Runtime version 3.1 (x64) and install the 6.0 version.
5. Restart the Universal Orchestrator service (see Start the Universal Orchestrator Service on page 2953).

5.2.1.2 Create Service Accounts for the Universal Orchestrator

The Keyfactor Universal Orchestrator makes use of up to two service accounts to allow it to communicate with the Keyfactor Command server. These two service accounts work together to transfer information from the Universal Orchestrator to the Keyfactor Command server. The two service accounts can be thought of as players on two sides of a fence, with the service account that the Universal Orchestrator runs as lobbing information over the fence to the service account that communicates with the Keyfactor Command server side to catch and hand to the Keyfactor Command server. Below, these are referred to as the Universal Orchestrator service account and the Keyfactor Command connect service account.

The service accounts need to be created prior to installation of the Universal Orchestrator software (except as noted below for installations on Linux), and the person installing the Universal Orchestrator software needs to know the domain (if applicable), username and password of each service account.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Universal Orchestrator Service Account

Your choice of service account may vary depending on the operating system on which you are installing the orchestrator:

Universal Orchestrator on a Windows Server

When the Universal Orchestrator is installed on Windows, you may use either the built-in Network Service account or a custom service account as the Universal Orchestrator service account. Keyfactor recommends using the default of Network Service unless you have a need to use a custom service account. If you choose to use a custom service account, it may be a standard Active Directory service account, an Active Directory group managed service account (gMSA), or a local machine account. Of the custom service account choices, an Active Directory account is more typically used unless the machine is not domain-joined. If you use an Active Directory service account, it needs to be a service account in the forest in which the Universal Orchestrator is installed. This is not necessarily the same forest as the forest in which the Keyfactor Command server is installed. The Universal Orchestrator on Windows has several possible roles, and the choice of service account may vary depending on these roles:

SSL Management

If your Universal Orchestrator SSL discovery and monitoring, you may choose to run the orchestrator as the built-in Network Service account or as a custom service account.

CA Management

If your Universal Orchestrator will be providing certificate synchronization from a remote CA, the Universal Orchestrator service account needs to be able to read the CA(s) in the forest in which the Universal Orchestrator is installed to retrieve certificates and templates from them. When the Universal Orchestrator is used in this configuration, this is typically a forest other than the forest in which the Keyfactor Command server is installed. For domain-joined CAs, you would typically use an Active Directory service account in the remote forest (the forest where the Universal Orchestrator is installed). For a non-domain-joined CA, you may use a local account created on the CA as the Universal Orchestrator service account instead of a domain account.

Custom Extensions

Keyfactor offers several publicly available custom extensions for the Universal Orchestrator in the Keyfactor GitHub. Many of these will operate correctly with a Universal Orchestrator service account running as Network Service, but some may require a custom account. Check the specific documentation for each custom extension for more information:

https://keyfactor.github.io/integrations-catalog/content/orchestrator

The Keyfactor Orchestrator Service on the server on which the Universal Orchestrator is installed runs as the Universal Orchestrator service account. This service account requires local “Log on as a service” permissions; this permission is granted automatically during installation.

Universal Orchestrator on a Linux Server

For the purposes of this documentation, it is assumed that Linux machines will be non-domain joined and will use a local account to run the Universal Orchestrator.

For Linux systems, Keyfactor recommends running the service as an account other than root. The default Universal Orchestrator service account of *keyfactor-orchestrator* will be created automatically during the install if the *force* option is used. If you prefer not to use the *force* option, you may create a local service account before running the installation script.

Universal Orchestrator in a Linux Container

This service account is not relevant for the orchestrator run in a container, since the container build is self-contained.

Keyfactor Command Connect Service Account

For the Keyfactor Command connect service account, the service account you use depends on the identity provider you're using:

- If you're using Active Directory as an identity provider, a standard Active Directory service account in the primary Keyfactor Command server forest is used. Group managed service accounts are not supported in this role.

  Tip: If the Universal Orchestrator is installed on Windows in the same forest as the Keyfactor Command server, the same Active Directory service account may be used as both the Universal Orchestrator service account and the Keyfactor Command connect service account, if desired.

- If you're using an identity provider other than Active Directory, a client (not user) in the identity provider is used. The client should be configured with a secret and have *Client authentication* and *Service account roles* enabled (see Service Accounts on page 2730). The user installing the orchestrator will need the client ID and secret.

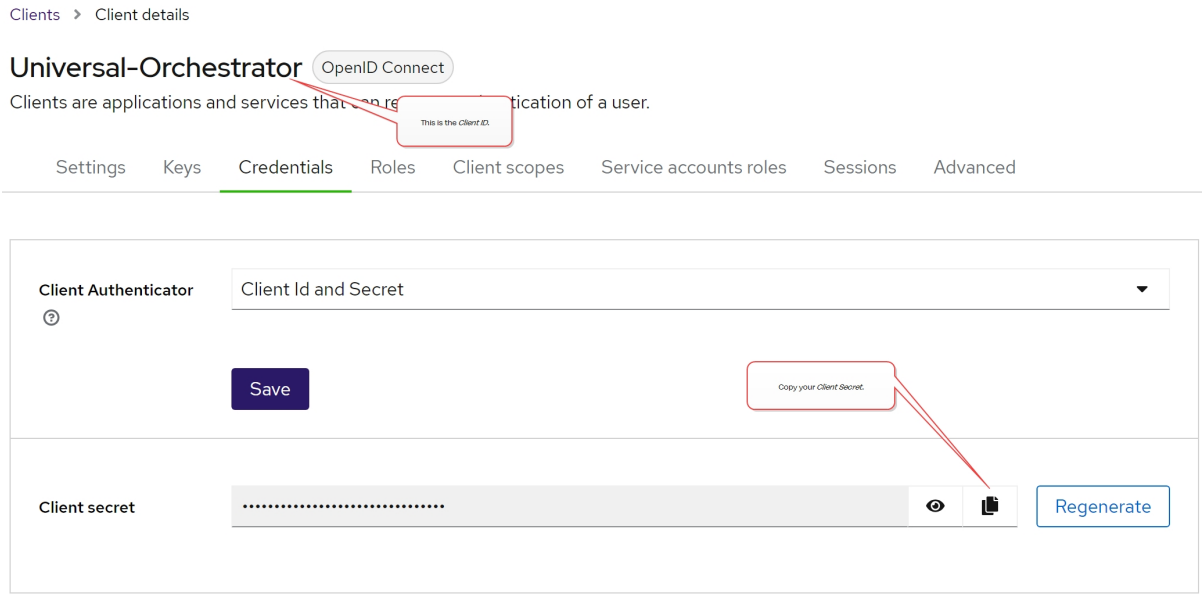


Figure 568: Client Secret for Orchestrator Client in Keyfactor Identity Provider

This service account appears in the Management Portal Orchestrator Management grid as the Identity for the Universal Orchestrator.

Permissions

The user installing the orchestrator must have the SeBackupPrivilege and SeRestorePrivilege rights on the Keyfactor Universal Orchestrator server. Normally, administrators are granted these permissions by default, but you should confirm the permissions prior to starting the install. These permissions can be set through Group Policy or Local Security Policy, and can be found under *Local Policies\User Rights Assignment* as *Back up files and directories* and *Restore files and directories*.

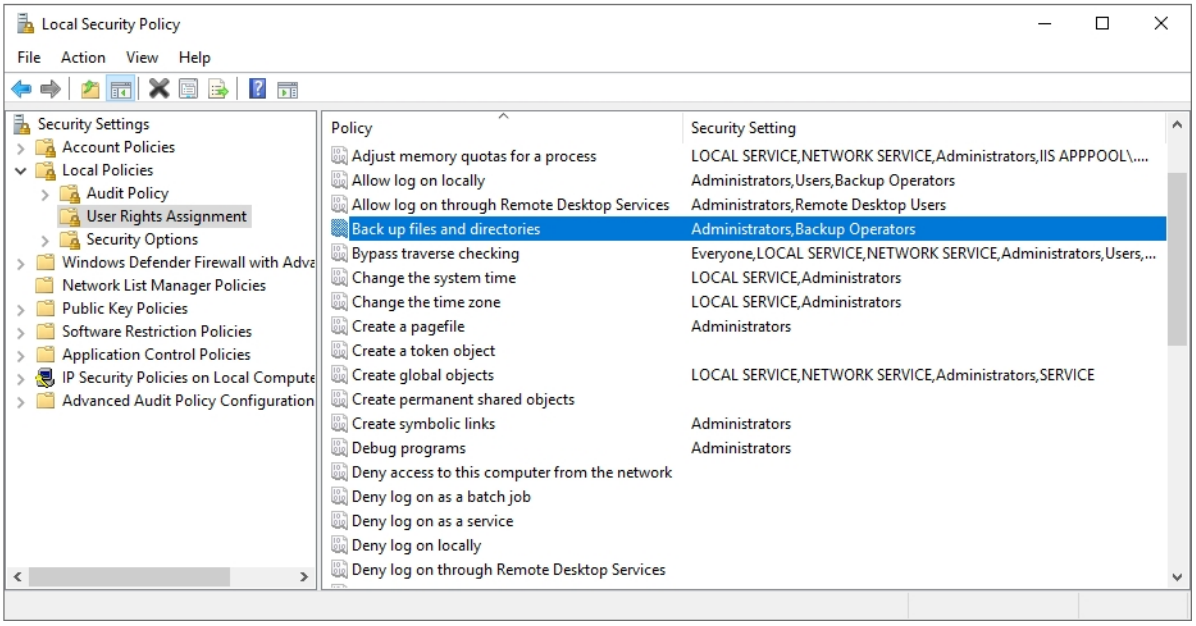


Figure 569: Local Security Policy

For more information on this from Microsoft, see:

> https://docs.microsoft.com/en-us/windows/win32/api/userenv/nf-userenv-loaduserprofilea#remarks

5.2.1.3 Configure Certificate Root Trust for the Universal Orchestrator

Keyfactor recommends using HTTPS to secure the channel between each Keyfactor Universal Orchestrator and the Keyfactor Command server(s). This requires an SSL certificate configured in IIS on the Keyfactor Command server(s). This certificate can either be a publicly-rooted certificate (e.g. from DigiCert, Entrust, etc.), or one issued from a private certificate authority (CA). If your Keyfactor Command server is using a publicly rooted certificate, the orchestrator server may already trust the certificate root for this certificate. However, if you have opted to use an internally-generated certificate, your orchestrator server may not trust this certificate. In order to use HTTPS for communications between the orchestrator and the Keyfactor Command server with a certificate generated from a private CA, you may need to import the certificate chain for the certificate into either the local machine certificate store on the orchestrator server on Windows or the root certificate store on Linux.

Note: The CRL(s) for the Keyfactor Command certificate need to be available to the orchestrator (see Troubleshooting on page 3003).

Installations on Windows Servers

If the public key infrastructure (PKI) that issued the certificate has only a root CA, the root certificate from this CA must be installed in the Trusted Root Certification Authorities store under Local Computer on the orchestrator server. If the PKI that issued the certificate has both a root and issuing CA, the root certificate must be installed in the Trusted Root Certification Authorities store under Local Computer on the orchestrator server and the issuing CA certificate must be installed in the Intermediate Certification Authorities store under Local Computer on the orchestrator server.

Installations on Linux Servers and in Linux Containers

The location of the OpenSSL trusted root store varies depending on your Linux implementation. The root certificate must be installed in the appropriate location for the operating system before beginning the installation.

5.2.1.4 Grant the Orchestrator Service Account Permissions on the CAs

This step only needs to be completed if you plan to use the Keyfactor Universal Orchestrator for remote Microsoft CA synchronization.

In order for the Universal Orchestrator to be able to synchronize certificates from the remote Microsoft CA(s) to the Keyfactor Command database, the Universal Orchestrator service account—the identity under which the orchestrator in the remote forest runs—must have permissions to read the CA database(s) in the remote forest.

In the management console for each CA that the orchestrator will interact with, open the properties for the CA and grant the service account that the orchestrator runs as (see Create Service Accounts for the Universal Orchestrator on page 2884) read permissions before continuing.

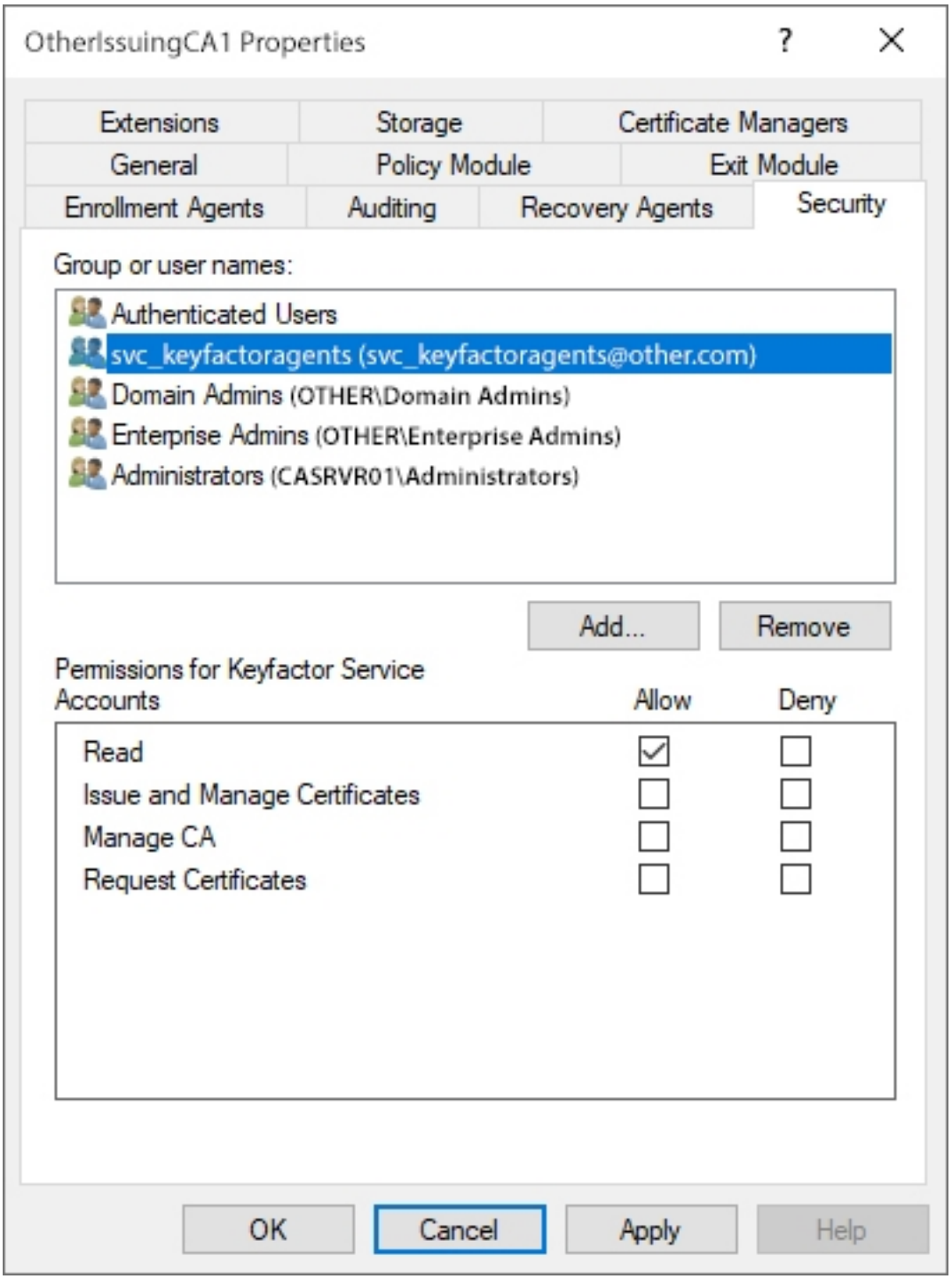


Figure 570: CA Permissions

5.2.1.5 Acquire a Certificate for Client Certificate Authentication (Optional)

The Keyfactor Universal Orchestrator supports client certificate authentication to allow you to authenticate via client certificates from individual orchestrator machines to either a centralized proxy, such as a network load balancer, which would in turn authenticate to the Keyfactor Command server using either a username and password or client ID and secret that was stored securely on the proxy or another client certificate, or directly using IIS on the Keyfactor Command to manage the certificate authentication and Active Directory to manage the mapping of client certificates to service accounts. The proxy approach allows orchestrator credentials to be assigned and managed outside the Active Directory forest in which Keyfactor Command is installed. The web proxy's job is to confirm the validity of the certificate and to provide Active Directory or an identity provider other than Active Directory credentials known to Keyfactor Command (if configured in this manner). Typically the proxy would be configured to accept all certificates issued from a given PKI implementation—even a PKI that is unknown to the Keyfactor Command Active Directory forest—thus delegating orchestrator access control to that PKI. For more information, see:

- Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC on page 3023
- Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication with Certificates Stored in Active Directory on page 3035

Important: The Universal Orchestrator supports automated client certificate renewal using an extension point interface on the orchestrator that can be implemented by the end-user. The custom extension will generate a CSR with private key and submit the CSR to Keyfactor Command for enrollment. Keyfactor Command will return the certificate to the orchestrator, which will pair it with its private key and use that certificate for authentication. See Register a Client Certificate Renewal Extension on page 2961 for more information.

Note: Client certificate authentication is not supported when using the Universal Orchestrator installed in a Linux container (see Install the Universal Orchestrator in a Linux Container on page 2922).

There are several situations in which using certificate authentication for the Universal Orchestrator may be helpful, including:

- Scale—To allow orchestrator numbers to scale (e.g. the IoT case) where it isn't practical to have a unique Active Directory account for each orchestrator.
- Untrusted Environments—To support environments (e.g. a "hostile" network) where policy doesn't allow the password for an Active Directory account to be stored on the orchestrator.

The certificate that the Universal Orchestrator uses for authentication needs:

- An extended key usage (EKU) of Client Authentication

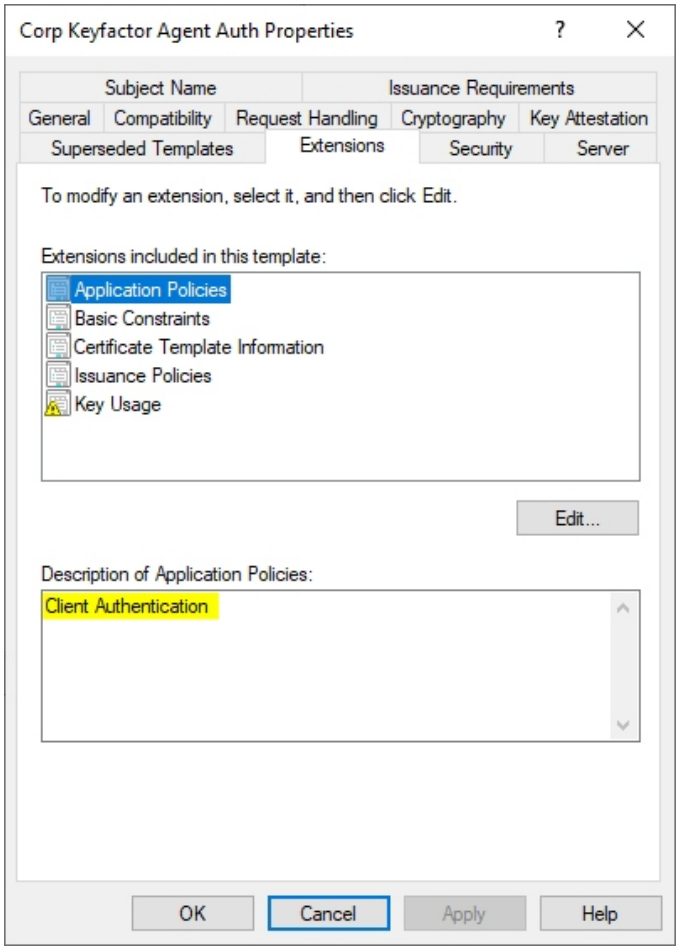


Figure 571: Microsoft Certificate Template Application Policies for Client Authentication Certificate

- A key usage that includes Digital Signature

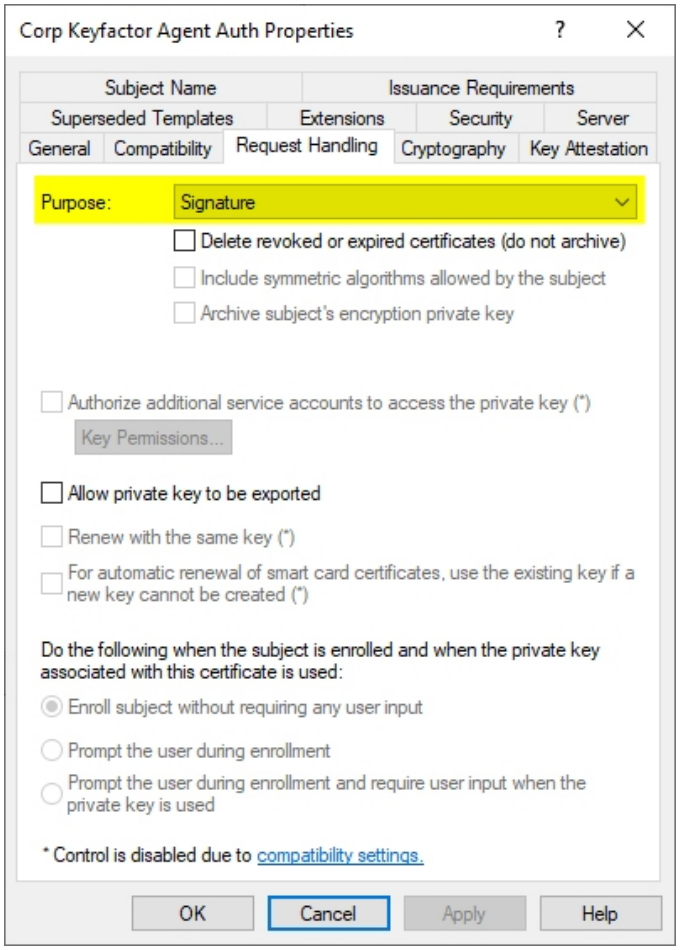


Figure 572: Microsoft Certificate Template Request Handling for Client Authentication Certificate

On Windows servers, the certificate may be referenced either as a PKCS12 file stored in the file system or may be place either in the local machine's personal store (*My*), or, if you opt to run the Universal Orchestrator service as a domain service account rather than the default of *Network Service*, in the personal store of the Universal Orchestrator service account user. If you opt to place the certificate in the local machine store, you need to grant the service account under which the Universal Orchestrator service will run (including *Network Service* if you will use this option) read permissions to the private key of the certificate. If you opt to place the certificate in the personal store of the Universal Orchestrator service account user, it also needs to be placed in the personal store of the user running the installation for the duration of the installation to allow it to be read during initial configuration. It may be removed from the installing user's store after installation is complete.

On Linux servers, the certificate is referenced as a PKCS12 file stored in the file system.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

To acquire a certificate for use by the Universal Orchestrator using a Microsoft CA, first create a template using the appropriate configurations as described above and make it available for enrollment on the CA from which you will request the certificate. The simplest way to acquire a certificate as a PKCS12 file for either Linux or Windows use is with PFX enrollment in Keyfactor Command. There are multiple ways to acquire a certificate and place it in the machine store on the Windows server where the Universal Orchestrator will be installed, including:

- Enroll through the Microsoft certificates MMC.
- Generate a CSR through the Microsoft certificates MMC and take the CSR to Keyfactor Command to issue a certificate using the CSR enrollment option in the Keyfactor Command Management Portal. You will need to return to the Microsoft certificates MMC to marry the certificate with the private key.
- Enroll for a certificate through Keyfactor Command using the PFX enrollment method and deploy it to the certificate store using an already installed Universal Orchestrator managing the store as an IIS store.
- Enroll using the command-line certreq command with a request.inf file on the Universal Orchestrator server.

Several of the above methods can also be used if you opt to enroll into the Universal Orchestrator service account user's personal store, though this option requires a few extra steps.

To enroll for a certificate using the certificates MMC into the local machine store:

1. On the Universal Orchestrator machine, do one of following:
    - Using the GUI:
        a. Open an empty instance of the Microsoft Management Console (MMC).
        b. Choose **File->Add/Remove Snap-in...**.
        c. In the *Available snap-ins* column, highlight **Certificates** and click **Add**.
        d. In the Certificates snap-in popup, choose the radio button for Computer account, click **Next**, accept the default of Local computer, and click **Finish**.
        e. Click **OK** to close the Add or Remove Snap-ins dialog.
    - Using the command line:
        a. Open a command prompt using the "Run as administrator" option.
        b. Within the command prompt type the following to open the certificates MMC:

            ```
            certlm.msc
            ```

2. Drill down to the Personal folder under **Certificates** for the Local Computer, right-click, and choose **All Tasks->Request New Certificate...**.

3. Follow the certificate enrollment wizard, selecting the template you created or identified for use for this purpose, and providing any required information.

4. When the enrollment completes, locate the certificate in the Personal store (you may need to refresh), highlight it, and choose **All Tasks->Manage Private Keys...**.

5. In the Permissions for private keys dialog, click **Add**, add the Universal Orchestrator service account—the account under which the Universal Orchestrator is running (created as per Create Service Accounts for the Universal Orchestrator on page 2884)—and grant that service account **Read** but not **Full control** permissions. Click **OK** to save.

5.2.1.6 Upgrading the Universal Orchestrator

There are two possible paths for upgrading from an earlier implementation of the Keyfactor Universal Orchestrator to a newer implementation:

- If your newer orchestrator will be installed in the same path as the older orchestrator, you may install the newer orchestrator over the older orchestrator using the *-Force* (Windows) or *--force* (Linux) option to overwrite the existing implementation.
- You may uninstall the older implementation using the provided uninstall script (uninstall.ps1 on Windows or uninstall.sh on Linux) and install the newer version using the standard installation steps (see Install the Universal Orchestrator on Windows on page 2898 or Install the Universal Orchestrator on a Linux Server on page 2912).

If you have an existing instance of the Keyfactor Windows Orchestrator and wish to migrate to the Keyfactor Universal Orchestrator, you may either install the two orchestrators side-by-side and then uninstall the Keyfactor Windows Orchestrator or uninstall the Keyfactor Windows Orchestrator and then install the Keyfactor Universal Orchestrator.

Important: Before following any of these upgrade paths, be sure to save off a copy of any custom extensions for the Keyfactor Universal Orchestrator (found in C:\Program Files\Keyfactor\Keyfactor Orchestrator\extensions by default) or plugins for the Keyfactor Windows Orchestrator (found in C:\Program Files\Keyfactor\Keyfactor Windows Orchestrator\plugins by default).

Keyfactor's suggested upgrade process is:

1. Review your installed orchestrator and gather this information:

    - Is this a Windows Orchestrator or a Universal Orchestrator?

      There are a number of ways to tell the difference. For example, the default install directory for the Universal Orchestrator is *Keyfactor Orchestrator* which the default install directory for the Windows Orchestrator is *Keyfactor Windows Orchestrator*. The Universal Orchestrator has several subdirectories, including an *extensions* directory. The Keyfactor Windows Orchestrator has only one subdirectory—*plugins*—by default.

- What user account is being used to run the orchestrator service?

  To check this, you can open the Windows Services tool (services.msc), look for the *Keyfactor Orchestrator Service*, and check the account that's configured as the *Log On As*.
- If this installation is on Windows, is the user account being used to run the orchestrator service a group managed service account (gMSA)?
- What type of authentication is being used to make the connection to Keyfactor Command?

  To check this, you can open the appsettings.json configuration file, which is found in the following location by default:

  Windows: C:\Program Files\Keyfactor\Keyfactor Orchestrator\configuration\appsettings.json

  Linux: /opt/keyfactor/orchestrator/configuration/appsettings.json

  If you're using client certificate authentication, the CertPath and AuthCertThumbprint fields will be populated. If you're using token authentication, the BearerTokenUrl and ClientId fields will be populated. If none of the aforementioned fields are populated, you're using Basic authentication.
- What user account, client ID, or other authentication information is being used to make the connection to Keyfactor Command?

  Check in Keyfactor Command for the Identity that the orchestrator indicates on the Orchestrator Management page.
- What secret information (user password, client secret, etc.) is used to make the connection to Keyfactor Command?

  This information cannot be retrieved from your existing installation. It is stored in an encrypted state.
- What plugins (Windows Orchestrator) or extensions (Universal Orchestrator) are you using?

  The plugins for the Keyfactor Windows Orchestrator are found in C:\Program Files\Keyfactor\Keyfactor Windows Orchestrator\plugins by default. The extensions for the Keyfactor Universal Orchestrator are found in C:\Program Files\Keyfactor\Keyfactor Orchestrator\extensions by default.

2. If you're using plugins or extensions:

   a. Before beginning the upgrade, save off a copy of any existing plugins or extensions in use.

   b. If you're using plugins or extensions from the Keyfactor Git Hub, check for the latest version:

      https://keyfactor.github.io/integrations-catalog/content/orchestrator

      Plugins for the Keyfactor Windows Orchestrator are not compatible with the Keyfactor Universal Orchestrator, so check for extensions that replace your plugins.

c. Review the documentation of each extension you will be using to determine if there are any changes needed to the certificate store type definition in Keyfactor Command. Only the following fields in a certificate store type that's in use may be edited:

   - Name
   - Short Name
   - Supported Job Types
   - Entry Parameters

   If changes to any other fields of the certificate store type are needed, you will need to create a new certificate store type.

d. Create new certificate store types or edit existing certificate store types as needed (see the previous step). If you're creating a new certificate store type to replace an existing one, it cannot have the same Short Name as an existing certificate store type. This means that you will most likely be using a non-standard Short Name for your extension (e.g. CitrixAdc2) and will need to modify the extension configuration to point it to the correct certificate store type. To do this, in the directory for your extension (the version that you will later copy into the upgraded orchestrator's directory), locate the manifest.json file and open it for editing. Change the existing capability to map to your new certificate store type(s). For example, CitrixAdc becomes CitrixAdc2:

```
{
   "extensions": {
      "Keyfactor.Orchestrators.Extensions.IOrchestratorJobExtension": {
         "CertStores.CitrixAdc2.Inventory": {
            "assemblypath": "Keyfactor.Extensions.Orchestrator.CitricAdc.dll",
            "TypeFullName": "Keyfactor.Extensions.Orchestrator.CitricAdc.Inventory"
         },
         "CertStores.CitrixAdc2.Management": {
            "assemblypath": "Keyfactor.Extensions.Orchestrator.CitricAdc.dll",
            "TypeFullName": "Keyfactor.Extensions.Orchestrator.CitricAdc.Management"
         }
      }
   }
}
```

3. Install the new Universal Orchestrator, either uninstalling the previous version or installing over the previous version using the -force option, following the standard installation steps (see Install the Universal Orchestrator on Windows on the next page, Install the Universal Orchestrator on a Linux Server on page 2912, or Install the Universal Orchestrator in a Linux Container on page 2922) and referencing the information you gathered in step 1. If you're installing in a container, you'll need to stage your selected extensions as part of the install.

4. If you're using extensions and didn't install in a container, copy your extensions into the extensions directory of the new orchestrator (see Installing Custom-Built Extensions on page 2940).

Restart the orchestrator service to pick up the extension changes.

5. In the Keyfactor Command Management Portal, review the orchestrator, confirm that the capabilities are as expected, and approve the orchestrator.

6. If you're using extensions and added a new certificate store type, you will need to recreate your certificate store. The certificate store type associated with a certificate store cannot be edited. Review your current certificate stores and recreate them with the new certificate store type, and then delete the versions with the old certificate store type.

7. Confirm that your certificate stores and/or SSL scanning are functioning as expected.

5.2.2 Install the Universal Orchestrator on Windows

To install the Keyfactor Universal Orchestrator on Windows, copy the zip file containing installation files to a temporary working directory on the Windows server and unzip it.

Note: In some instances, downloading a compressed file on Windows can cause the file to be marked as *blocked*. If you unzip a blocked file and proceed with the installation, the installation may fail with an error about missing files or dependencies (e.g. "Could not load file or assembly [filename] or one of its dependencies…"). Before beginning the installation, check the zip file *before* unzipping it to confirm that it is not blocked and unblock it if it is blocked.

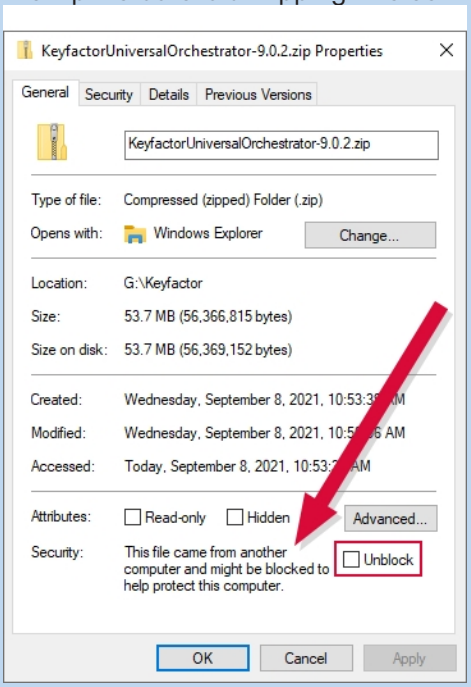


Figure 573: Installation Files Blocked after Download

To begin the installation:

1. On the Windows machine on which you wish to install the orchestrator, open a PowerShell window using the "Run as Administrator" option and change to change to *InstallationScripts* subdirectory under the temporary directory where you placed the installation files.

2. In the PowerShell window, select from the following commands to run based on the identity provider you're using for Keyfactor Command, the desired orchestrator service accounts, and the desired install experience to prepare for the install.

   In these examples, *credKeyfactor* is used for the Keyfactor Command connect service account that the orchestrator uses to connect to Keyfactor Command and *credService* is used for the Universal Orchestrator service account that the service runs as. Usernames should be given in DOMAIN\username format for Active Directory domain accounts or hostname\username format for local user accounts.

   - If you're using Active Directory as an identity provider, will be running the service as an Active Directory domain or local account rather than Network Service, and do not want to provide the usernames and passwords in the command, run the following commands to populate a variable with the user credentials for the Keyfactor Command connect service account (see Create Service Accounts for the Universal Orchestrator on page 2884) and populate a variable with the user credentials for the Universal Orchestrator service account:

     ```
     $credKeyfactor = Get-Credential
     $credService = Get-Credential
     ```

     Enter the appropriate username and password when prompted.

   - Active Directory as the identity provider, running the service as Network Service, and not providing the username and password for the Keyfactor Command connect service account in the command:

     ```
     $credKeyfactor = Get-Credential
     ```

   - Active Directory as the identity provider, running the service as Network Service, and providing the username and password for the Keyfactor Command connect service account in the command:

     ```
     $keyfactorUser = "DOMAIN\mykeyfactorconnectusername"
     $keyfactorPassword = "MySecurePassword"
     $secKeyfactorPassword = ConvertTo-SecureString $keyfactorPassword -AsPlainText
     -Force
     $credKeyfactor = New-Object System.Management.Automation.PSCredential
     ($keyfactorUser, $secKeyfactorPassword)
     ```

   - Active Directory as the identity provider, running the service as a domain or local account, and providing the username and password for the Keyfactor Command connect service account and the service account the orchestrator runs as in the command:

```
$serviceUser = "DOMAIN\myserviceusername"
$keyfactorUser = "DOMAIN\mykeyfactorconnectusername"
$keyfactorPassword = "MyFirstSecurePassword"
$servicePassword = "MySecondSecurePassword"
$secKeyfactorPassword = ConvertTo-SecureString $keyfactorPassword -AsPlainText
-Force
$secServicePassword = ConvertTo-SecureString $servicePassword -AsPlainText -
Force
$credKeyfactor = New-Object System.Management.Automation.PSCredential
($keyfactorUser, $secKeyfactorPassword)
$credService = New-Object System.Management.Automation.PSCredential
($serviceUser, $secServicePassword)
```

- Active Directory as the identity provider, running the service as a group managed service account (gMSA), and providing the username and password for the Keyfactor Command connect service account in the command:

```
$serviceUser = "DOMAIN\myGMSAserviceusername$"
$keyfactorUser = "DOMAIN\mykeyfactorconnectusername"
$keyfactorPassword = "MySecurePassword"
$secKeyfactorPassword = ConvertTo-SecureString $keyfactorPassword -AsPlainText
-Force
$credKeyfactor = New-Object System.Management.Automation.PSCredential
($keyfactorUser, $secKeyfactorPassword)
$credService = New-Object System.Management.Automation.PSCredential
($serviceUser,(New-Object System.Security.SecureString))
```

  Note: Group managed service accounts are not supported for use in making the connection to Keyfactor Command.

- An identity provider other than Active Directory and running the service as Network Service:

```
none
```

- An identity provider other than Active Directory, running the service as a domain or local account, and not providing the username and password for the service account the orchestrator runs as in the command:

```
$credService = Get-Credential
```

- An identity provider other than Active Directory, running the service as a domain or local account, and providing the username and password for the service account the orchestrator runs as in the command:

```
$serviceUser = "DOMAIN\myserviceusername"
$servicePassword = "MySecondSecurePassword"
$secServicePassword = ConvertTo-SecureString $servicePassword -AsPlainText -
Force
$credService = New-Object System.Management.Automation.PSCredential
($serviceUser, $secServicePassword)
```

Tip: In some cases, you may be using the same service account for both the Universal Orchestrator service account role and the Keyfactor Command connect service account role. If this is the case, you may use a single variable for both passwords in the next step.

3. In the PowerShell window, run the install.ps1 script using the following syntax to begin the installation:

-URL (Required)

This is the URL to the Agent Services endpoint on the Keyfactor Command server running the Keyfactor Command Agent Services role. If you installed all the Keyfactor Command server roles together, this is the URL for your Keyfactor Command server with /KeyfactorAgents after the server's IP or FQDN (e.g. https://keyfactor.keyexample.com/KeyfactorAgents). If you choose to use SSL to connect to the Keyfactor Command server, you'll need to enter a URL that contains a hostname that is found in the SSL certificate.

This parameter sets the local orchestrator application setting *AgentsServerUri* to the specified value.

This parameter is **required**.

Note: If you've opted to use client certificate authentication for the orchestrator, the value you use for the **URL** will vary depending on the method you select to implement client certificate authentication. You may choose to route client certificate authentication through a proxy (see Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC on page 3023), in which case you would use the proxy server name here (whatever name you're using to route traffic through the proxy). You may choose to publish client certificates to Active Directory and access the Keyfactor Command server directly (see Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication with Certificates Stored in Active Directory on page 3035), in which case you would use the Keyfactor Command server name here.

Tip: If your Keyfactor Command server was configured with an alternate virtual directory for the Keyfactor Command Agents Services endpoint, you will need to enter that in the URL rather than /KeyfactorAgents.

Client Authentication Parameters (Required)

The Keyfactor Universal Orchestrator supports authenticating to the Keyfactor Command server using Basic authentication, token authentication, or client certificate authentication. The method you choose depends in part on the identity provider in use for the Keyfactor Command the orchestrator will be communicating with. If you're using Active Directory as an identity provider, you may choose Basic authentication or client certificate authentication. If you're using an identity provider other than Active Directory, you may choose token authentication or client certificate authentication.

When you configure the orchestrator with Basic authentication (*WebCredential*), you provide a username and password as a PSCredential object. With token authentication (*BearerTokenUrl*), you provide a client ID and secret that allows the orchestrator to acquire a bearer token. With client certificate authentication (either *ClientCertificateThumbprint* or *ClientCertificate* and *ClientCertificatePassword*), the orchestrator uses a client certificate to authenticate to either a proxy or IIS on the Keyfactor Command server. You cannot configure multiple types of authentication together.

One of the following authentication methods is **required**:

- Basic Authentication: *WebCredential*
- Token Authentication: *BearerTokenUrl*, *ClientId*, *ClientSecret*, and *TokenLifetime*
- Client Certificate Authentication: *ClientCertificate* and *ClientCertificatePassword*
- Client Certificate Authentication: *ClientCertificateThumbprint*

Important: Choosing a client certificate authentication method for the orchestrator may require additional configuration on your Keyfactor Command server. For more information, see Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication with Certificates Stored in Active Directory on page 3035, Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC on page 3023, and Install the Keyfactor Command Components on the Keyfactor Command Server(s) on page 2780.

Tip: For information about rotating passwords and client authentication certificates, see Change Service Account Passwords on page 2954.

-WebCredential (Basic Authentication)

This is the credential object of the Keyfactor Command connect service account that the orchestrator uses to communicate with Keyfactor Command that you created as per Create Service Accounts for the Universal Orchestrator on page 2884. It is provided as a PSCredential object.

This parameter is **required** if Basic authentication will be used.

This parameter cannot be used in conjunction with the *BearerTokenURL*, *ClientCertificateThumbprint*, *ClientCertificate*, or *ClientCertificatePassword* parameter.

-BearerTokenURL (Token Authentication)

Specifying this parameter causes the installation to be done using token authentication for the connection to Keyfactor Command.

This parameter **requires** that *TokenLifetime*, *ClientId*, and *ClientSecret* also be specified.

This parameter is **required** if token authentication will be used.

This parameter cannot be used in conjunction with the *WebCredential*, *ClientCertificateThumbprint*, *ClientCertificate*, or *ClientCertificatePassword* parameter.

-ClientId (Token Authentication)

This parameter is used to specify the ID of the identity provider client that should be used to authenticate the session when *BearerTokenUrl* authentication is used (see Create Service Accounts for the Universal Orchestrator on page 2884).

This parameter **requires** that *TokenLifetime* and *ClientSecret* also be specified.

This parameter is only supported if the *BearerTokenUrl* parameter is specified.

-ClientSecret (Token Authentication)

This parameter is used to specify the secret of the identity provider client that should be used to authenticate the session when *BearerTokenUrl* authentication is used.

This parameter **requires** that *TokenLifetime* and *ClientId* also be specified.

This parameter is only supported if the *BearerTokenUrl* parameter is specified.

-TokenLifetime (Token Authentication)

The number of seconds for which the bearer token is valid. The *TokenLifetime* should be set to the same value as the Keyfactor Command *Cookie Expiration* (see Authentication Tab on page 2787). For example, if the Keyfactor Command *CookieExpiration* is 5 minutes, the *TokenLifetime* should be 300 seconds.

The *Cookie Expiration* value determines the length of time the authentication cookie is considered valid. After half of the setting's duration, Keyfactor Command will attempt to use a refresh token to update the cookie. If this fails, the orchestrator's session will be terminated.

This parameter **requires** that *ClientId* and *ClientSecret* also be specified.

This parameter is only supported if the *BearerTokenUrl* parameter is specified.

-ClientCertificate (Client Certificate Authentication)

The path and file name on the orchestrator of a PKCS12 file containing the client authentication certificate used to authenticate to Keyfactor Command created as per Acquire a Certificate for Client Certificate Authentication (Optional) on page 2891. The certificate must have a Client Authentication EKU.

The account under which the Universal Orchestrator service will run (see -ServiceCredential on page 2907) needs read and write permissions on the PKCS12 file you specify with this parameter.

This parameter **requires** that *ClientCertificatePassword* also be specified.

You may specify either the thumbprint of the certificate with the *ClientCertificateThumbprint* parameter or specify a path and password to a PKCS12 file containing the certificate on the orchestrator using *ClientCertificate* and *ClientCertificatePassword*. You do not need to specify both a thumbprint and a PKCS12 file; if you do, the certificate stores will take precedence.

Specifying this parameter sets the local orchestrator application setting *CertPath* to the specified value.

This parameter cannot be used in conjunction with the *BearerTokenURL* or *WebCredential* parameter.

-ClientCertificatePassword (Client Certificate Authentication)

The password for the PKCS12 file specified with the ClientCertificate parameter.

Specifying this parameter **requires** that *ClientCertificate* also be specified.

This parameter cannot be used in conjunction with the *BearerTokenURL* or *WebCredential* parameter.

-ClientCertificateThumbprint (Client Certificate Authentication)

The thumbprint of the client authentication certificate used to authenticate to Keyfactor Command created as per Acquire a Certificate for Client Certificate Authentication (Optional) on page 2891. The certificate must have a Client Authentication EKU, have a private key readable by the account under which the Universal Orchestrator service will run (see -ServiceCredential on page 2907), and be located in either the orchestrator local machine's personal certificate store (*My*) or the Universal Orchestrator service account user's (see -ServiceCredential on page 2907) personal certificate store. If the certificate is stored in the local machine's store, the Universal Orchestrator service account user must be granted permissions to read the private key of the certificate (see the final steps under Acquire a Certificate for Client Certificate Authentication (Optional) on page 2891).

You may specify either the thumbprint of the certificate with the *ClientCertificateThumbprint* parameter or specify a path and password to a PKCS12 file containing the certificate on the orchestrator using *ClientCertificate* and *ClientCertificatePassword*. You do not need to specify both a thumbprint and a PKCS12 file; if you do, the certificate stores will take precedence.

Specifying this parameter sets the local orchestrator application setting *AuthCertThumbprint* to the specified value.

This parameter cannot be used in conjunction with the *BearerTokenURL* or *WebCredential* parameter.

-Audience

This parameter is used to specify an audience value to be included in token requests delivered to the identity provider when using an identity provider other than Active Directory.

-Capabilities

This parameter is used to specify the capabilities the orchestrator will support if a capability set other than the default set is desired. Supported options are:

- all

  All the capabilities supported by the orchestrator will be enabled and reported to Keyfactor Command.

- none

  The orchestrator will be installed with no capabilities and will not be registered with Keyfactor Command. This is primarily used for implementations that will support only custom capabilities (see Installing Custom-Built Extensions on page 2940 and Configuring Script-Based Certificate Store Jobs on page 2946).

- ssl

  Only the SSL discovery and monitoring capability will be enabled and reported to Keyfactor Command.

If the *InPlace* parameter is specified, this parameter must be set to *all*.

If this parameter is not specified, the default set of capabilities for the orchestrator will be used. For the Universal Orchestrator, the default capability set is *IIS*, *CA* and *LOG* (log fetching).

One installation of the orchestrator can be enabled with multiple capabilities to perform more than one function, but there are best practices for locating orchestrators that should be considered. For example, Keyfactor recommends against performing the SSL discovery and monitoring function using an orchestrator installed on the main Keyfactor Command server due to the resource requirements of this function and against using the same orchestrator for the SSL function and other functions, again due to the resource requirements. The CA management

function is typically used on remote servers and not collocated with other orchestrator functions.

-Destination

This parameter specifies a location in which to install the orchestrator that is other than the default. The default installation location is:

```
C:\Program Files\Keyfactor\Keyfactor Orchestrator
```

This parameter cannot be used in conjunction with the *InPlace* parameter.

-Force

Specifying this parameter causes the installation to warn and continue on certain potential problems, including:

- A service with either the default service name or the service name specified with the *ServiceSuffix* parameter already exists. The service will be overwritten if *Force* is specified.
- Either the default installation location or the location specified with the *Location* parameter is not empty. The install will occur to the specified or default location anyway and files may be overwritten if *Force* is specified.

If this parameter is not specified and any of these problems are encountered, the installation will terminate prematurely.

-InPlace

This parameter is used to indicate that the installation should occur in the current directory where the install files are located and no files should be copied to another location on the machine.

This parameter cannot be used in conjunction with the *Destination* parameter. This parameter is only supported if the *Capabilities* parameter is set to *all*.

-NoRevocationCheck

This parameter is used to indicate that the revocation status (CRL) of the SSL certificate on the Keyfactor Command server should not be checked when connecting to Keyfactor Command.

Specifying this parameter sets the local orchestrator application setting *CheckServerCertificateRevocation* to false. The default for this parameter is *true* (CRL checking will be done).

-NoService

This parameter is used to indicate that no Windows service should be created. The orchestrator will be installed but will need to be started manually or added as a service at a later time.

This parameter cannot be used in conjunction with the *ServiceSuffix* or *ServiceCredential* parameter.

-OrchestratorName

Specifying this parameter allows you to override the name the orchestrator would by default use to register itself in Keyfactor Command.

Specifying this parameter sets the local orchestrator application setting *OrchestratorName* to the specified value.

By default, the orchestrator uses the value of the COMPUTERNAME environment variable for the orchestrator's name.

-ServiceCredential

This is the credential object of the Universal Orchestrator service account the orchestrator service will run as (see Create Service Accounts for the Universal Orchestrator on page 2884. It is provided as a PSCredential object.

This parameter cannot be used in conjunction with the *NoService* parameter.

If this parameter is not specified, the built-in Network Service account will be used.

-ServiceSuffix

This parameter is used to add a suffix to the root service name of *KeyfactorOrchestrator* (e.g. *Instance1* for a resulting service name of *KeyfactorOrchestrator-Instance1*). This is used primarily for implementations where the orchestrator will be installed multiple times on the same server.

This parameter cannot be used in conjunction with the *NoService* parameter.

If this parameter is not specified, the default service name of *KeyfactorOrchestrator-Default* will be used—with a display name of *Keyfactor Orchestrator Service (Default)*.

-Scope

This parameter is used to specify one or more scopes that should be included in token requests delivered to the identity provider when using an identity provider other than Active Directory. Multiple scopes should be separated by spaces.

-Source

Specify this parameter to point to a directory containing the installation files other than the directory in which the install.ps1 file is found. This parameter is used primarily if a copy of the install.ps1 file is made in an alternate directory, updated with some customizations, and then used for installation without being copied back to the directory where the remaining installation files are located.

Installation example with expected output using Basic authentication and Network Service to run the local service:

```
$keyfactorUser = "KEYEXAMPLE\svc_kyforch1"
$keyfactorPassword = "MySecurePassword123!"
$secKeyfactorPassword = ConvertTo-SecureString $keyfactorPassword -AsPlainText -Force
$credKeyfactor = New-Object System.Management.Automation.PSCredential ($keyfactorUser, $secKey-
factorPassword)

.\install.ps1 -URL https://keyfactor.keyexample.com/KeyfactorAgents -WebCredential $credKeyfactor
-OrchestratorName websrvr42.keyexample.com -Capabilities all

Copying files
Setting configuration data
Installing Windows Service
Granting necessary file permissions to NT AUTHORITY\NETWORK SERVICE for configuration file
Starting service KeyfactorOrchestrator-Default
```

Installation example with expected output using Basic authentication and a standard Active Directory service account to run the local service:

```
$serviceUser = "KEYEXAMPLE\svc_kyforch1"
$keyfactorUser = "KEYEXAMPLE\svc_kyforch2"
$servicePassword = "MyFirstSecurePassword123!"
$keyfactorPassword = "MySecondSecurePassword456#"
$secServicePassword = ConvertTo-SecureString $servicePassword -AsPlainText -Force
$secKeyfactorPassword = ConvertTo-SecureString $keyfactorPassword -AsPlainText -Force
$credService = New-Object System.Management.Automation.PSCredential ($serviceUser, $secSer-
vicePassword)
$credKeyfactor = New-Object System.Management.Automation.PSCredential ($keyfactorUser,
$secKeyfactorPassword)

.\install.ps1 -URL https://keyfactor.keyexample.com/KeyfactorAgents -WebCredential $credKeyfactor
-ServiceCredential $credService -OrchestratorName websrvr42-IIS.keyexample.com -Capabilities all

Copying files
```

```
Setting configuration data
Installing Windows Service
Granting necessary file permissions to KEYEXAMPLE\svc_kyforch1 for configuration file
Granting Log on as a Service permission to KEYEXAMPLE\svc_kyforch1
Starting service KeyfactorOrchestrator-Default
```

Installation example with expected output using Basic authentication and an Active Directory gMSA to run the local service:

```
$serviceUser = "KEYEXAMPLE\GMSA_kyforch$"
$keyfactorUser = "KEYEXAMPLE\svc_kyforch"
$keyfactorPassword = "MySecurePassword123!"
$secKeyfactorPassword = ConvertTo-SecureString $keyfactorPassword -AsPlainText -Force
$credService = New-Object System.Management.Automation.PSCredential ($serviceUser,(New-Object
System.Security.SecureString))
$credKeyfactor = New-Object System.Management.Automation.PSCredential ($keyfactorUser, $secKey-
factorPassword)

.\install.ps1 -URL https://keyfactor.keyexample.com/KeyfactorAgents -WebCredential $credKeyfactor
-ServiceCredential $credService -OrchestratorName websrvr42-IIS.keyexample.com -Capabilities all

Copying files
Setting configuration data
Installing Windows Service
Granting necessary file permissions to KEYEXAMPLE\GMSA_kyforch$ for configuration file
Granting Log on as a Service permission to KEYEXAMPLE\GMSA_kyforch$
Starting service KeyfactorOrchestrator-Default
```

Important: Prior to using a gMSA in the installation, you need to have installed the account on the Universal Orchestrator server using the *Install-ADServiceAccount* PowerShell command. For example:

```
Install-ADServiceAccount -Identity GMSA_kyforch$
```

This requires the *Active Directory module for Windows PowerShell*, which is installed as a feature as part of the *Remote Server Administrator Tools*.

Installation example with expected output using token authentication and Network Service to run the local service:

```
.\install.ps1 -URL https://keyfactor.keyexample.com/KeyfactorAgents -BearerTokenUrl https://appsr-
vr18.keyexample.com:1443/realms/Keyfactor/protocol/openid-connect/token -TokenLifetime 300 -
```

```
ClientId Universal-Orchestrator -ClientSecret m1aE6RErW5cezSPmv0PJcFdFp152HFqK -OrchestratorName
websrvr42-UO.keyexample.com -Capabilities all

Copying files
Setting configuration data
Installing Windows Service
Granting necessary file permissions to NT AUTHORITY\NETWORK SERVICE for configuration file
Starting service KeyfactorOrchestrator-Default
```

Installation example with expected output using token authentication and a local account on the machine to run the local service:

```
$serviceUser = "websrvr42\kyforch"
$servicePassword = "MySecurePassword123!"
$secServicePassword = ConvertTo-SecureString $servicePassword -AsPlainText -Force
$credService = New-Object System.Management.Automation.PSCredential ($serviceUser, $secSer-
vicePassword)

.\install.ps1 -URL https://keyfactor.keyexample.com/KeyfactorAgents -BearerTokenUrl https://appsr-
vr18.keyexample.com:1443/realms/Keyfactor/protocol/openid-connect/token -TokenLifetime 300 -
ClientId Universal-Orchestrator -ClientSecret m1aE6RErW5cezSPmv0PJcFdFp152HFqK -ServiceCredential
$credService -OrchestratorName websrvr42-UO.keyexample.com -Capabilities all

Copying files
Setting configuration data
Installing Windows Service
Granting necessary file permissions to websrvr42\kyforch for configuration file
Granting Log on as a Service permission to websrvr42\kyforch
Starting service KeyfactorOrchestrator-Default
```

Installation example with expected output using client certificate authentication with the certificate stored in the local machine store:

```
$serviceUser = "KEYEXAMPLE\svc_kyforch"
$servicePassword = "MySecurePassword123!"
$secServicePassword = ConvertTo-SecureString $servicePassword -AsPlainText -Force
$credService = New-Object System.Management.Automation.PSCredential ($serviceUser, $secSer-
vicePassword)

.\install.ps1 -URL https://keyfactor.keyexample.com/KeyfactorAgents -ClientCertificateThumbprint
29b21df7403b4afe6daf44762e5c47fb73c07ce7 -ServiceCredential $credService -OrchestratorName
websrvr42-IIS.keyexample.com -Capabilities all
```

```
Copying files
Setting configuration data
Installing Windows Service
Granting necessary file permissions to KEYEXAMPLE\svc_kyforch for configuration file
Granting Log on as a Service permission to KEYEXAMPLE\svc_kyforch
Starting service KeyfactorOrchestrator-Default
```

Tip: The client certificate authentication example shown here references a certificate stored in the local machine store. Because of this, the service account that will run the Universal Orchestrator service needs to be granted permissions to read the private key of the certificate before the installation is run. If the certificate had been acquired into the Universal Orchestrator service account user's personal store rather than the local machine store, the step of granting private key read permissions would not have been necessary.

Installation example with expected output using client certificate authentication with the certificate stored as a file:

```
.\install.ps1 -URL https://keyfactor.keyexample.com/KeyfactorAgents -ClientCertificate
C:\Certs\kyforch.pfx -ClientCertificatePassword MySecurePassword123! -OrchestratorName websrvr42-
IIS.keyexample.com -Capabilities all

Copying files
Setting configuration data
Installing Windows Service
Granting necessary file permissions to KEYEXAMPLE\svc_kyforch for configuration file
Granting Log on as a Service permission to KEYEXAMPLE\svc_kyforch
Starting service KeyfactorOrchestrator-Default
```

Tip: The client certificate authentication example shown here does not use the -ServiceCredential parameter. This will cause the Universal Orchestrator service to run as Network Service. If you prefer to run the service as a domain service account, you will need to include the -ServiceCredential parameter and specify the PSCredential value for the service credentials appropriately, as shown in the previous examples.
Network Service will need to be granted read and write permissions on the PFX file before the script is executed.

4. Review the output from the installation to confirm that no errors have occurred.

The script creates a directory, C:\Program Files\Keyfactor\Keyfactor Orchestrator by default, and places the orchestrator files in this directory. Log files are found in C:\Program Files\Keyfactor\Keyfactor Orchestrator\logs by default, though this is configurable (see Configure Logging for the Universal Orchestrator on page 2950).

The orchestrator service, by default given a display name of *Keyfactor Orchestrator Service (Default)*, should be automatically started at the conclusion of the install and configured to restart on reboot unless you have selected the *NoService* parameter.

Tip: Once the installation of the orchestrator is complete, you need to use the Keyfactor Command Management Portal to approve the orchestrator and configure certificate stores or SSL jobs:

- Approving or Disapproving Orchestrators on page 500
- Certificate Store Operations on page 413
- SSL Discovery on page 453

If you've opted to enable remote CA management for the orchestrator, further configuration is needed (see Configure the Universal Orchestrator for Remote CA Management on page 2938).

5.2.3 Install the Universal Orchestrator on a Linux Server

To install the Keyfactor Universal Orchestrator on a Linux server, copy the zip file containing installation files to a temporary working directory on the Linux server and unzip it.

To begin the installation:

1. On the Linux machine on which you wish to install the orchestrator, in a command shell change to *InstallationScripts* subdirectory under the temporary directory where you placed the installation files.

2. Use the chmod command to make the install.sh script file executable. The file ships in a non-executable state to avoid accidental execution. For example:

   ```
   sudo chmod +x install.sh
   ```

3. In the command shell, run the install.sh script as root using the following parameters to begin the installation:

   #### --url (Required)

   This is the URL to the Agent Services endpoint on the Keyfactor Command server running the Keyfactor Command Agent Services role, which is installed as part of the Keyfactor Command Services role. If you installed all the Keyfactor Command server roles together, this is the URL for your Keyfactor Command server with /KeyfactorAgents after the server's IP or FQDN (e.g. https://keyfactor.keyexample.com/KeyfactorAgents). If you choose to use SSL to connect to the Keyfactor Command server, you'll need to enter a URL that contains a hostname that is found in the SSL certificate.

This parameter sets the local orchestrator application setting *AgentsServerUri* to the specified value.

This parameter is **required**.

Tip: If your Keyfactor Command server was configured with an alternate virtual directory for the Keyfactor Command Agents Services endpoint, you will need to enter that in the URL rather than /KeyfactorAgents.

Client Authentication Parameters (Required)

The Keyfactor Universal Orchestrator supports authenticating to the Keyfactor Command server using Basic authentication, token authentication, or client certificate authentication. The method you choose depends in part on the identity provider in use for the Keyfactor Command the orchestrator will be communicating with. If you're using Active Directory as an identity provider, you may choose Basic authentication or client certificate authentication. If you're using an identity provider other than Active Directory, you may choose token authentication or client certificate authentication.

When you configure the orchestrator with Basic authentication (*username* and *password*), you provide a username and password. With token authentication (*bearer-token-url*, *client_id*, and *client_secret*), you provide a client ID and secret that allows the orchestrator to acquire a bearer token. With client certificate authentication (*client-auth-certificate* and *client-auth-certificate-password*), the orchestrator uses a client certificate to authenticate to either a proxy or IIS on the Keyfactor Command server. You cannot configure multiple types of authentication together.

One of the following authentication methods is **required**:

- Basic Authentication: *username* and *password*
- Token Authentication: *bearer-token-url*, *client_id*, *client_secret*, and *token_lifetime*
- Client Certificate Authentication: *client-auth-certificate* and *client-auth-certificate-password*

Important: Choosing to use client certificate authentication for the orchestrator may require additional configuration on your Keyfactor Command server. For more information, see Install the Keyfactor Command Components on the Keyfactor Command Server(s) on page 2780 in the *Keyfactor Command Server Installation Guide* and Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication with Certificates Stored in Active Directory on page 3035, Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC on page 3023.

Tip: For information about rotating passwords and client authentication certificates, see Change Service Account Passwords on page 2954.

--username (Basic Authentication)

This is the Keyfactor Command connect service account that the orchestrator uses to communicate with Keyfactor Command that you created as per Create Service Accounts for the Universal Orchestrator on page 2884. It may be entered either as username@domain (e.g. svc_kyforch@keyexample.com) or DOMAIN\\username (e.g. KEYEXAMPLE\\svc_kyforch).

This parameter is **required** if Basic authentication will be used.

This parameter cannot be used in conjunction with the *bearer-token-url* or *client-auth-certificate* and *client-auth-certificate-password* parameters.

--password (Basic Authentication)

This is the password for the Keyfactor Command connect service account that the orchestrator uses to communicate with Keyfactor Command specified with the *username* parameter.

Important: The password for the Keyfactor Command connect service account is stored in clear text in the orchestratorsecrets.json file in the configuration directory under the installation directory for the orchestrator. By default, this file is granted read/write permissions for the Universal Orchestrator service account running the service on the Linux machine (*keyfactor-orchestrator* by default) and no permissions for any other users. Access to this file should be strictly controlled.
If you prefer to avoid the use of a password in a file, consider using client certificate authentication.

This parameter is **required** if the *username* parameter is specified.

This parameter cannot be used in conjunction with the *bearer-token-url* or *client-auth-certificate* and *client-auth-certificate-password* parameters.

Important: Your password may be preserved in command history and may be visible on the process listing when providing a password using this parameter. See the *--secret-file-path* and --secret-std-in parameters for alternatives.

--bearer-token-url (Token Authentication)

Specifying this parameter causes the installation to be done using token authentication for the connection to Keyfactor Command. Set this to the URL of the token endpoint for your identity provider. For example:

```
https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token
```

For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716).

This parameter **requires** that *token-lifetime*, *client-id*, and *client-secret* also be specified.

This parameter is **required** if token authentication will be used.

This parameter cannot be used in conjunction with the *username* and *password* or *client-auth-certificate* and *client-auth-certificate-password* parameters.

--client-id (Token Authentication)

This parameter is used to specify the ID of the identity provider client that should be used to authenticate the session when *bearer-token-url* authentication is used (see Create Service Accounts for the Universal Orchestrator on page 2884).

This parameter **requires** that *token-lifetime* and *client-secret* also be specified.

This parameter is only supported if the *bearer-token-url* parameter is specified.

--client-secret (Token Authentication)

This parameter is used to specify the secret of the Keyfactor Identity Provider client that should be used to authenticate the session when *bearer-token-url* authentication is used.

Important: The client secret for the Keyfactor Command connect service account is stored in clear text in the orchestratorsecrets.json file in the configuration directory under the installation directory for the orchestrator. By default, this file is granted read/write permissions for the Universal Orchestrator service account running the service on the Linux machine (*keyfactor-orchestrator* by default) and no permissions for any other users. Access to this file should be strictly controlled.
If you prefer to avoid the use of a password in a file, consider using client certificate authentication.

This parameter **requires** that *token-lifetime* and *client-id* also be specified.

This parameter is only supported if the *bearer-token-url* parameter is specified.

Important: Your secret may be preserved in command history and may be visible on the process listing when providing a secret using this parameter. See the *--secret-file-path* and --secret-std-in parameters for alternatives.

--token-lifetime (Token Authentication)

The number of seconds for which the bearer token is valid. The *token-lifetime* should be set to the same value as the Keyfactor Command *CookieExpiration*. For example, if the Keyfactor Command *CookieExpiration* is 5 minutes, the *token-lifetime* should be 300 seconds.

The *Cookie Expiration* value determines the length of time the authentication cookie is considered valid. After half of the setting's duration, Keyfactor Command will attempt to use a refresh token to update the cookie. If this fails, the orchestrator's session will be terminated.

This parameter **requires** that *client-id* and *client-secret* also be specified.

This parameter is only supported if the *bearer-token-url* parameter is specified.

--client-auth-certificate (Client Certificate Authentication)

The path and file name on the orchestrator of a PKCS12 file containing the client authentication certificate used to authenticate to Keyfactor Command created as per Acquire a Certificate for Client Certificate Authentication (Optional) on page 2891. The certificate must have a Client Authentication EKU.

The account under which the Universal Orchestrator service will run (see --service-user on page 2920) needs read and write permissions on the PKCS12 file you specify with this parameter.

This parameter **requires** that *client-auth-certificate-password* also be specified.

Specifying this parameter sets the local orchestrator application setting *CertPath* to the specified value.

This parameter cannot be used in conjunction with the *bearer-token-url* or *username* and *password* parameters.

--client-auth-certificate-password (Client Certificate Authentication)

The password for the PKCS12 file specified with the *client-auth-certificate* parameter.

Specifying this parameter **requires** that *client-auth-certificate* also be specified.

This parameter cannot be used in conjunction with the *bearer-token-url* or *username* parameters.

> ⚠ **Important:** Your password may be preserved in command history and may be visible on the process listing when providing a password using this parameter. See the *--secret-file-path* and --secret-std-in parameters for alternatives.

--secret-file-path (All Authentication Types)

This parameter specifies a path and filename to provide a plain text secret for the Keyfactor Command connect service account that the orchestrator uses to communicate with Keyfactor Command. For example:

```
sudo ./install.sh --secret-file-path /opt/apps/my_secret_file [other parameters here]
```

This parameter can be used with the *username*, *client-auth-certificate*, or *client-id* parameter to provide the authentication secret from a file rather than the command line to avoid storing it in command history.

This parameter cannot be used in conjunction with the *password*, *client_secret*, or *client-auth-certificate-password* parameter.

Tip: Be sure to delete your secret file at the conclusion of the installation.

--secret-std-in (All Authentication Types)

This parameter allows you to provide a plain text secret via standard in for the Keyfactor Command connect service account that the orchestrator uses to communicate with Keyfactor Command. For example:

```
echo "MySuperSecretPassword" | sudo ./install.sh --secret-std-in [other parameters here]
```

This parameter can be used with the *username*, *client-auth-certificate*, or *client-id* parameter to provide the authentication secret from a file rather than the command line to avoid storing it in command history.

This parameter cannot be used in conjunction with the *password*, *client_secret*, or *client-auth-certificate-password* parameter.

--audience

This parameter is used to specify an audience value to be included in token requests delivered to the identity provider when using an identity provider other than Active Directory.

--capabilities

This parameter is used to specify the capabilities the orchestrator will support if a capability set other than the default set is desired. Supported options are:

- all

  All the capabilities supported by the orchestrator will be enabled and reported to Keyfactor Command.

- none

  The orchestrator will be installed with no capabilities and will not be registered with Keyfactor Command. This is primarily used for implementations that will support only custom capabilities (see Installing Custom-Built Extensions on page 2940 and Configuring Script-Based Certificate Store Jobs on page 2946).

- ssl

  Only the SSL discovery and monitoring capability will be enabled and reported to Keyfactor Command.

If the *in-place* parameter is specified, this parameter must be set to *all*.

If this parameter is not specified, the default set of capabilities for the orchestrator will be used. For the Linux orchestrator, the default capability set is *LOG* (log fetching).

> ⚠ **Important:** The Linux orchestrator does not support the CA (remote CA management) or IIS (Windows server certificate store) capabilities due to the Windows-specific nature of the authentication requirements for these methods.

--destination

This parameter specifies a location in which to install the orchestrator that is other than the default. The default installation location is:

```
/opt/keyfactor/orchestrator
```

This parameter cannot be used in conjunction with the *in-place* parameter.

--force, -f

Specifying this parameter causes the installation to warn and continue on certain potential problems, including:

- The local Universal Orchestrator service account does not exist. The default user will be created if *force* is specified.
- The local application settings (appsettings.json) file does not exist. A new one will be created if *force* is specified.
- A service with either the default service name or the service name specified with the *service-suffix* parameter already exists. The service will be overwritten if *force* is specified.
- Either the default installation location or the location specified with the *location* para-

meter is not empty. The install will occur to the specified or default location anyway and files may be overwritten if *force* is specified.

If this parameter is not specified and any of these problems are encountered, the installation will terminate prematurely. See also the *what-if* parameter.

--in-place

This parameter is used to indicate that the installation should occur in the current directory where the install files are located and no files should be copied to another location on the machine.

This parameter cannot be used in conjunction with the *destination* parameter. This parameter is only supported if the *capabilities* parameter is set to *all*.

--no-revocation-check

This parameter is used to indicate that the revocation status (CRL) of the SSL certificate on the Keyfactor Command server should not be checked when connecting to Keyfactor Command.

Specifying this parameter sets the local orchestrator application setting *CheckServerCertificateRevocation* to false. The default for this parameter is *true* (CRL checking will be done).

--no-service

This parameter is used to indicate that no service should be created and added to the server's service control manager. The orchestrator will be installed but will need to be started manually or added to the server's service control manager manually.

This parameter cannot be used in conjunction with the *service-suffix* or *service-user* parameter.

--orchestrator-name

Specifying this parameter allows you to override the name the orchestrator would by default use to register itself in Keyfactor Command.

Specifying this parameter sets the local orchestrator application setting *OrchestratorName* to the specified value.

By default, the orchestrator uses the results from a hostname lookup for the orchestrator's name.

--service-suffix

This parameter is used to add a suffix to the root service name of *keyfactor-orchestrator* (e.g. *instance1* for a resulting service name of *keyfactor-orchestrator-instance1*). This is used primarily for implementations where the orchestrator will be installed multiple times on the same

server.

This parameter cannot be used in conjunction with the *no-service* parameter.

If this parameter is not specified, the default service name of *keyfactor-orchestrator-default* will be used.

--service-user

This is the local Linux Universal Orchestrator service account that the service will run as (see Create Service Accounts for the Universal Orchestrator on page 2884. It should be entered as just the user name. Entry of a password for this service account is not required. You may either create this account prior to running the installation script (or use an existing account) or use the *force* parameter to generate the account automatically during the installation process.

This parameter cannot be used in conjunction with the *no-service* parameter.

If this parameter is not specified, the default service account name of *keyfactor-orchestrator* will be used.

--scope

This parameter is used to specify one or more scopes that should be included in token requests delivered to the identity provider when using an identity provider other than Active Directory. Multiple scopes should be separated by spaces.

--source

Specify this parameter to point to a directory containing the installation files other than the directory in which the install.sh file is found. This parameter is used primarily if a copy of the install.sh file is made in an alternate directory, updated with some customizations, and then used for installation without being copied back to the directory where the remaining installation files are located.

--verbose, -v

Specify this parameter to output verbose installation messages.

--what-if

This parameter is used to test the installation command without actually installing in order to see any errors that might arise and correct them before installing.

Installation example with expected output using Basic authentication (the password for the svc_kyforch service account is saved in my_password_file):

```
vi my_password_file
```

```
sudo ./install.sh --url https://keyfactor.keyexample.com/KeyfactorAgents --username svc_
kyforch@keyexample.com --secret-file-path my_password_file --orchestrator-name appsrvr16-
ssl.keyexample.com --capabilities all --force

Creating user keyfactor-orchestrator
Copying files from /tmp/KeyfactorOrchestrator to /opt/keyfactor/orchestrator
Saving app settings
Setting file permissions
Installing systemd service keyfactor-orchestrator-default
Created symlink /etc/systemd/system/multi-user.target.wants/keyfactor-orchestrator-default.ser-
vice → /etc/systemd/system/keyfactor-orchestrator-default.service.
Starting systemd service keyfactor-orchestrator-default
```

Installation example with expected output using token authentication (the secret for the client is provided at standard in):

```
echo "WcHlahyku6wmD0a6rjOXClrkz0Jw9sGh" | sudo ./install.sh --url https://key-
factor.keyexample.com/KeyfactorAgents --bearer-token-url https://appsr-
vr18.keyexample.com:1443/realms/Keyfactor/protocol/openid-connect/token --token-lifetime 300 --
client-id Universal-Orchestrator --secret-std-in --orchestrator-name appsrvr16-ssl.keyexample.com
--capabilities all --force

Creating user keyfactor-orchestrator
Copying files from /tmp/KeyfactorOrchestrator to /opt/keyfactor/orchestrator
Setting file permissions and saving app settings
Installing systemd service keyfactor-orchestrator-default
Created symlink /etc/systemd/system/multi-user.target.wants/keyfactor-orchestrator-default.ser-
vice → /etc/systemd/system/keyfactor-orchestrator-default.service.
Starting systemd service keyfactor-orchestrator-default
```

Installation example with expected output using client certificate authentication (the password for the client certificate is saved in cert_password_file):

```
vi cert_password_file

sudo ./install.sh --url https://keyfactor.keyexample.com/KeyfactorAgents --client-auth-certi-
ficate /opt/certs/kyforch.p12 --secret-file-path cert_password_file --orchestrator-name
appsrvr16-ssl.keyexample.com --capabilities all --force

Creating user keyfactor-orchestrator
Copying files from /tmp/KeyfactorOrchestrator to /opt/keyfactor/orchestrator
Saving app settings
Setting file permissions
```

```
Installing systemd service keyfactor-orchestrator-default
Created symlink /etc/systemd/system/multi-user.target.wants/keyfactor-orchestrator-default.ser-
vice → /etc/systemd/system/keyfactor-orchestrator-default.service.
Starting systemd service keyfactor-orchestrator-default
```

4. Review the output from the installation to confirm that no errors have occurred.

The script creates a directory, /opt/keyfactor/orchestrator by default, and places the orchestrator files in this directory. Log files are found in /opt/keyfactor/orchestrator/logs by default, though this is configurable (see Configure Logging for the Universal Orchestrator on page 2950).

The orchestrator service, by default named keyfactor-orchestrator-default.service, should be automatically started at the conclusion of the install and configured to restart on reboot unless you have selected the *no-service* parameter.

Tip: Once the installation of the orchestrator is complete, you need to use the Keyfactor CommandManagement Portal to approve the orchestrator and configure certificate stores or SSL jobs:

- Approving or Disapproving Orchestrators on page 500
- Certificate Store Operations on page 413
- SSL Discovery on page 453

5.2.4 Install the Universal Orchestrator in a Linux Container

When the Keyfactor Universal Orchestrator runs in a Linux container, it is typically installed in a containerization solution that sits on top of a Linux server or set of servers. There are a wide variety of containerization solutions for multiple operating systems. This document covers deploying the container to either Docker or Kubernetes on Linux.

The artifactory for the Universal Orchestrator images can be found here:

keyfactor.jfrog.io/con-develop-us-engineering/command/

Check with your Keyfactor Customer Success Manager for credentials.

Two different images are available, depending on the functionality you are looking for:

- universal-orchestrator

  This image has no built-in functionality and is designed to be used with custom extensions.
- universal-orchestrator-ssl

  This image provides the SSL capability to provide support for SSL discovery and monitoring.

Docker

If you plan to use Docker, you may find it helpful to first run the Universal Orchestrator in the foreground so that it will output log messages to assist in troubleshooting.

Tip: If your Docker implementation hasn't been configured to inject your DNS server(s) into running containers, you may wish to do this so that the Universal Orchestrator will be able to do name resolution. To do this, on the Linux server(s) where you are running Docker, create or update the **/etc/docker/daemon.json** file, and add an entry similar to the following:

```
{"dns": ["DNS_IP_Address_1", "DNS_IP_Address_2"] }
```

To install the Universal Orchestrator in a Linux container and start the container using compose:

1. Create a directory from which you will run the Docker container (e.g. /opt/kyf_uo).
2. Select a Universal Orchestrator image, and from your Docker host, retrieve the Universal Orchestrator image from the artifactory with commands similar to the following (using credentials provided to you by Keyfactor; the password is saved in my_password.txt):

   ```
   cat my_password.txt | docker login keyfactor.jfrog.io --username username --
   password-stdin


   docker pull keyfactor.jfrog.io/con-develop-us-engineering/command/universal-
   orchestrator-ssl:11.0
   ```

   Important: Remove the my_password.txt file when complete.

3. Create a Docker compose file (compose.yaml) in the directory for your Docker container similar to the following, using the inputs as per Table 854: Linux Container Parameters, referencing the artifictory you pulled, selecting the appropriate authentication mechanism for your environment, and any additional volume mounts (see Custom Extensions on page 2925). The fields highlighted in red below indicate fields that need to be edited or that you may wish to edit.

   Important: When editing the file, be sure to preserve the indenting exactly as found. YAML requires a very specific file layout to function. If the indenting (multiples of two spaces) or layout is incorrect, you will receive an error when trying to install.

   ```
   services:
     universal-orchestrator:
       image: keyfactor.jfrog.io/con-develop-us-engineering/command/universal-orchestrator-ssl:11.0
       container_name: universal_orchestrator_1
       environment:
   ```

```
      COMMAND_AGENTS_URL: https://keyfactor.keyexample.com/KeyfactorAgents
      ORCHESTRATOR_NAME: appsrvr19-UO-1

      # Uncomment the next two lines to use Active Directory
      #USERNAME: KEYEXAMPLE\svc_kyforch
      #PASSWORD: MySuperSecretPassword

      # Uncomment the next two lines to use OAuth
      #BEARER_TOKEN_URL: https://appsrvr18.keyexample.com:1443/realms/Keyfactor/protocol/openid-
connect/token
      #TOKEN_LIFETIME: 300
      #CLIENTID: Universal-Orchestrator
      #CLIENT_SECRET: Client-Secret-from-Keyfactor-IdP
    volumes:
      - /etc/ssl/certs:/etc/ssl/certs:ro
```

Important: The password or secret for the Keyfactor Command connect service account is stored in clear text in this compose file. Access to this file should be strictly controlled.

4. Set the permissions on the *compose.yaml* file such that the file is owned by root and readable only by root (this assumes your Docker daemon is running as root, which is typical). For example:

```
sudo chown root:root compose.yaml
```

```
sudo chmod 400 compose.yaml
```

Tip: If you need to make edits to the compose file, you will need to make the file writable again. For example:

```
sudo chmod 600 compose.yaml
```

5. Execute the following command to install and run the container in the foreground:

```
sudo docker compose up
```

Press CTRL-C to stop it if it's running in the foreground. You can instead run it in the background by adding the -d flag like so, but it can sometimes be helpful to run it in the foreground initially so that you can easily review the log output live:

```
sudo docker compose up -d
```

Tip: To stop and start the container again after installation is complete, use the following commands:

```
sudo docker compose stop

sudo docker compose start
```

Or:

```
sudo docker compose restart
```

If you need to delete the container and try the install again, use this command:

```
sudo docker compose down
```

To review logs generated from the container, identify the container ID or name with this command:

```
sudo docker container ls
```

Then use the following command to output the current log (with the optional --follow to make output continuous):

```
sudo docker container logs [--follow] [container ID or name]
```

Custom Extensions

To use custom extensions with the orchestrators (see Installing Custom-Built Extensions on page 2940), you can either build them into a custom-built orchestrator image or reference them as external volume mounts. The latter works well for quick testing in a development environment, but you'll probably want to use a custom-built orchestrator image for a production deployment.

To run the orchestrator referencing an extension as an external volume mount:

1. Create a directory on your Linux server to host the extension(s) you wish to use and copy each extension you wish to use into this directory. For example:

```
/opt/kyf_uo/exts/f5-ext

/opt/kyf_uo/exts/citrix-ext
```

Make note of whether the extension documentation indicates whether the files in the extension need to exist within a subdirectory or whether they should be placed in the root of the directory you're creating (e.g. f5-ext).

2. Follow the instructions above, but modify the volumes section of your docker compose file to include the extension(s). For example (where /app/extensions is the path within the image where the extensions will live):

```
[See beginning above]
    volumes:
      - /etc/ssl/certs:/etc/ssl/certs:ro
      - /opt/kyf_uo/exts/f5-ext:/app/extensions/f5-ext
      - /opt/kyf_uo/exts/citrix-ext:/app/extensions/citrix-ext
```

To create a custom build of the orchestrator referencing extensions:

1. Create a directory on your Linux server to host the extension(s) you wish to use and copy each extension you wish to use into this directory. This should be a subdirectory of the directory in which you will build your custom orchestrator image. For example:

```
/opt/kyf_uo/exts/f5-ext


/opt/kyf_uo/exts/citrix-ext
```

   Make note of whether the extension documentation indicates whether the files in the extension need to exist within a subdirectory or whether they should be placed in the root of the directory you're creating (e.g. f5-ext).

2. In the directory above the extension directory (e.g. /opt/kyf_uo), create a file called *Dockerfile* and open it for editing. The entries in this file will vary depending on the extension(s) you wish to include in your build. Check the documentation for the specific extension for more information. The following example includes two extensions:

```
FROM keyfactor.jfrog.io/con-develop-us-engineering/command/universal-orchestrator:11.0
WORKDIR "/app/extensions/f5-ext"
COPY ./ext/f5-ext/ ./
WORKDIR "/app/extensions/citrix-ext"
COPY ./ext/citrix-ext/ ./
WORKDIR "/app"
```

   This build script sets the source for the artifactory and then changes directories within the image to the f5-ext directory. It copies the contents of the ext/f5-ext subdirectory under the current working directory on the host to the current directory in the image. It then repeats this change directory and copy for the Citrix Netscaler extension. Finally, it changes directory back

to the /app directory within the image before leaving the build. This last step is important to be sure the image will later function as expected.

3. Build your custom image by executing the following command from the directory in which your *Dockerfile* is located (where custom-uo-image is the name you give to your image):

```
docker build -t custom-uo-image .
```

4. Create your compose file as above, but referencing your custom image like so:

```
services:
  universal-orchestrator:
    image: custom-uo-image
    container_name: universal_orchestrator_f5_ns
[see remainder above]
```

5. Complete the install as above.

Kubernetes

To install the Universal Orchestrator in a Linux container and start the container using Kubernetes:

1. Create a directory from which you will run the container (e.g. /opt/kyf_uo).

2. Create a secret in Kubernetes for the credentials that the orchestrator(s) will use to authenticate to Keyfactor Command (see Create Service Accounts for the Universal Orchestrator on page 2884). For example, for Active Directory authentication:

```
echo -n 'keyexample\svc_kyforch' > ./username

echo -n 'MySuperSecretPassword' > ./password

kubectl create secret generic uo-credentials --from-file=./username --from-
file=./password
```

   For authentication with an identity provider other than Active Directory:

```
echo -n 'Universal-Orchestrator' > ./clientid

echo -n 'Client-Secret-from-Keyfactor-IdP' > ./clientsecret

kubectl create secret generic uo-credentials --from-file=./clientid --from-
file=./clientsecret
```

Important: The password or secret for the Keyfactor Command connect service account is stored in clear text in the "password" or "clientsecret" file. Be sure to delete it after the Kubernetes secret has been created.

3. Create a secret in Kubernetes for the credentials you will use to authenticate to the Keyfactor artifactory. For example:

```
kubectl create secret docker-registry keyregcred --docker-server=keyfactor.jfrog.io
--docker-username=MyUsername --docker-password=MySuperSecretPassword --docker-
email=my.email@my-domain.com
```

4. On your Kubernetes server, create a configmap containing CA root certificates, including the chain certificates for the SSL certificate on the Keyfactor Command server (see Configure Certificate Root Trust for the Universal Orchestrator on page 2888). For example:

```
kubectl create configmap ca-roots --from-file=/etc/ssl/certs/ca-certificates.crt
```

Note: The standard path to the trusted root store will vary depending on your Linux implementation.

5. Create a Kubernetes deployment file (e.g. uo_ssl.yaml) in the directory for your Kubernetes container similar to the following, using the inputs as per Table 854: Linux Container Parameters, referencing the artifictory for the image you wish to install, selecting the appropriate authentication mechanism for your environment, and any additional volume mounts. The fields highlighted in red below indicate fields that need to be edited or that you may wish to edit.

Important: When editing the file, be sure to preserve the indenting exactly as found. YAML requires a very specific file layout to function. If the indenting (multiples of two spaces) or layout is incorrect, you will receive an error when trying to install.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  # Give the pod a unique name if you plan to deploy more than one orchestrator to the same Kuber-
netes server or cluster
  name: keyfactor-uo-1
  labels:
    app.kubernetes.io/name: keyfactor-universal-orchestrator
    app.kubernetes.io/instance: ssl-1
    app.kubernetes.io/version: "11.0"
spec:
```

```
  # The universal orchestrator should not have replicas; instead use many different deployments
with different names if horizontal scaling is needed
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: keyfactor-universal-orchestrator
      app.kubernetes.io/instance: ssl-1
  template:
    metadata:
      labels:
        app.kubernetes.io/name: keyfactor-universal-orchestrator
        app.kubernetes.io/instance: ssl-1
    spec:
      initContainers:
        # The below two commented out blocks provide examples of adding custom extensions
        #- env:
        #  - name: EXTENSION_NAME
        #    value: citrix-adc-orchestrator
        #  - name: EXTENSION_VERSION
        #    value: 2.0.0
        #  - name: INSTALL_PATH
        #    value: /app/extensions/citrix-adc-orchestrator
        #  image: m8rmclarenkf/uo_extension_installer:1.0.5
        #  imagePullPolicy: IfNotPresent
        #  name: citrix-adc-orchestrator-installer
        #  volumeMounts:
        #  - mountPath: /app/extensions
        #    name: command-pv-claim
        #    readOnly: false
        #    subPath: ""
        #- env:
        #  - name: EXTENSION_NAME
        #    value: f5-rest-orchestrator
        #  - name: EXTENSION_VERSION
        #    value: 1.4.4
        #  - name: INSTALL_PATH
        #    value: /app/extensions/f5-rest-orchestrator
        #  image: m8rmclarenkf/uo_extension_installer:1.0.5
        #  imagePullPolicy: IfNotPresent
        #  name: f5-rest-orchestrator-installer
        #  volumeMounts:
        #  - mountPath: /app/extensions
        #    name: command-pv-claim
```

```
        #    readOnly: false
        #    subPath: ""
      containers:
        - name: keyfactor-universal-orchestrator-ssl-1
          # Uncomment the desired image
          image: "keyfactor.jfrog.io/con-develop-us-engineering/command/universal-orchestrator-
ssl:11.0"
          #image: "keyfactor.jfrog.io/con-develop-us-engineering/command/universal-orches-
trator:11.0"
          imagePullPolicy: IfNotPresent
          # Universal orchestrator environment
          env:
            # The below block is the URL of the orchestrator API on the Keyfactor Command server
            - name: COMMAND_AGENTS_URL
              value: https://keyfactor.keyexample.com/KeyfactorAgents
            # The below block is the name the orchestrator will use when registering with
Keyfactor Command
            - name: ORCHESTRATOR_NAME
              value: k8s-universal-orchestrator-ssl-1
            - name: LOG_LEVEL
              value: Info
            # Uncomment the next two blocks to use Active Directory authentication
            #- name: USERNAME
            #  valueFrom:
            #    secretKeyRef:
            #      name: uo-credentials
            #      key: username
            #- name: PASSWORD
            #  valueFrom:
            #    secretKeyRef:
            #      name: uo-credentials
            #      key: password
            # Uncomment the next four blocks to use OAuth authentication
            #- name: BEARER_TOKEN_URL
            #  value: https://appsrvr18.keyexample.com:1443/realms/Keyfactor/protocol/openid-
connect/token
            #- name: TOKEN_LIFETIME
            #  value: "300"
            #- name: CLIENTID
            #  valueFrom:
            #    secretKeyRef:
            #      name: uo-credentials
            #      key: clientid
```

```
        #- name: CLIENT_SECRET
        #  valueFrom:
        #    secretKeyRef:
        #      name: uo-credentials
        #      key: clientsecret
      volumeMounts:
        # Uncomment the next block if adding extensions
        #- mountPath: /app/extensions
        #  name: command-pv-claim
        #  readOnly: false
        #  subPath: ""
        - mountPath: /etc/ssl/certs/ca-certificates.crt
          name: root-ca
          readOnly: false
          subPath: ca-certificates.crt
  volumes:
    - configMap:
        items:
        - key: ca-certificates.crt
          path: ca-certificates.crt
        name: ca-roots
      name: root-ca
    # Uncomment the next block if adding extensions
    #- name: command-pv-claim
    #  emptyDir: {}
  imagePullSecrets:
    - name: keyregcred
```

6. Set the permissions on the deployment file such that the file is owned by root and readable only by root (this assumes your Kubernetes implementation is running as root, which is typical). For example:

```
sudo chown root:root uo_ssl.yaml

sudo chmod 400 uo_ssl.yaml
```

Tip: If you need to make edits to the compose file, you will need to make the file writable again. For example:

```
sudo chmod 600 uo_ssl.yaml
```

7. Execute the following command to install and run the container:

```
kubectl apply -f uo_ssl.yaml
```

Tip: To review logs generated from the container, identify the pod name with this command:

```
kubectl get pods
```

Then use the following command to output the current log:

```
kubectl logs [pod name] --follow
```

The optional follow parameter will continuously output the logs as they are generated until interrupted.

If you need to delete the container and try the install again, use this command:

```
kubectl delete -f uo_ssl.yaml
```

Table 854: Linux Container Parameters

Parameter	Description
AppSettings__Check-ServerCertificateRevocation	A Boolean that indicates whether the revocation status (CRL) of the SSL certificate on the Keyfactor Command server should be checked when connecting to Keyfactor Command (true) or not (false). The default is *true* (CRL checking will be done).
AUDIENCE	This parameter is used to specify an audience value to be included in token requests delivered to the identity provider when using an identity provider other than Active Directory.
BEARER_TOKEN_URL	**Required***. The URL of the token endpoint for your identity provider. For example: `https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token` For Keyfactor Identity Provider, this is included among the information that can be found on the OpenID Endpoint Configuration page, a link to which can be found on the Realm Settings page (see Configuring Keyfactor Identity Provider and Collecting Data for the Keyfactor Command Installation on page 2716). This parameter is **required** if you're using an identity provider other than Active Directory.

Parameter	Description
CLIENTID	**Required***. For implementations using an identity provider other than Active Directory, the ID of the identity provider client that should be used to authenticate the session (see Create Service Accounts for the Universal Orchestrator on page 2884). This parameter is **required** if you're using an identity provider other than Active Directory.
CLIENT_SECRET	**Required***. For implementations using an identity provider other than Active Directory, the secret of the identity provider client that should be used to authenticate the session. This parameter is **required** if you're using an identity provider other than Active Directory.
COMMAND_AGENTS_URL	**Required**. The URL of the Orchestrators API on the Keyfactor Command server. For example: `https://keyfactor.keyexample.com/KeyfactorAgents`
LOG_LEVEL	The logging level for the orchestrator. The default value is *Info*. Possible values are the same as those described in Configure Logging for the Universal Orchestrator on page 2950.
ORCHESTRATOR_NAME	The name the orchestrator uses to register itself with Keyfactor Command. By default, the container hostname is used, which is not ideal as this will create a new orchestrator entry with every container start. Although this parameter is not strictly required, Keyfactor strongly recommends using it. If you choose to uninstall and reinstall the orchestrator (using compose down), it is important to use the same orchestrator name for subsequent implementations so that Keyfactor Command will recognize the orchestrator when it is started again using compose up.
PASSWORD	**Required***. The password for the Keyfactor Command Connect Service Account if you're using Active Directory as an identity provider (see USERNAME). This parameter is **required** if you're using Active Directory as an identity provider.
SCOPE	This parameter is used to specify one or more scopes that should be included in token requests delivered to the identity provider when using an identity provider other than Active Directory. Multiple scopes should be separated by spaces.

Parameter	Description
TOKEN_LIFETIME	For implementations using an identity provider other than Active Directory, the number of seconds for which the bearer token is valid. This should be set to the same value as the Keyfactor Command *Cookie Expiration*. For example, if the Keyfactor Command *Cookie Expiration* is 5 minutes, the *TOKEN_LIFETIME* should be 300 seconds. The default value is 60. The *Cookie Expiration* value determines the length of time the authentication cookie is considered valid. After half of the setting's duration, Keyfactor Command will attempt to use a refresh token to update the cookie. If this fails, the orchestrator's session will be terminated.
USERNAME	**Required***. The username for service account used to connect to the Keyfactor Command server (see PASSWORD). This is the Keyfactor Command Connect Service Account described in Create Service Accounts for the Universal Orchestrator on page 2884 if you're using Active Directory as an identity provider. The orchestrator uses Basic Authentication to authenticate to Keyfactor Command. This parameter is **required** if you're using Active Directory as an identity provider.

Note: The Keyfactor Universal Orchestrator running in a container does not support client certificate authentication.

Tip: Once the installation of the orchestrator is complete, you need to use the Keyfactor CommandManagement Portal to approve the orchestrator and configure certificate stores or SSL jobs:

- Approving or Disapproving Orchestrators on page 500
- Certificate Store Operations on page 413
- SSL Discovery on page 453

5.2.5 Optional Configuration

Once the installation is complete, the Keyfactor Universal Orchestrator should be running and ready to communicate with the Keyfactor Command server. The initial installation allows the orchestrator to register itself with Keyfactor Command and run jobs of the capability types configured during installation (after being approved in the Keyfactor Command Management Portal) unless you selected the NoService parameter.

This section details some post-install configuration steps that may need to be completed for some capabilities and some optional settings.

Important: Synchronization for the remote CA functionality of the orchestrator will not begin until you complete the configuration by making the appropriate configuration changes in the Keyfactor Command Management Portal. See Orchestrator Management on page 496 in the *Keyfactor Command Reference Guide* for instructions on approving the orchestrator in the Keyfactor Command Management Portal on the *Orchestrators->Management* page and Adding or Modifying a CA Record on page 354 in the *Keyfactor Command Reference Guide* for instructions on configuring certificate and template synchronization for remote CAs on the *Locations->Certificate Authorities* page.

5.2.5.1 Configure Windows Targets for Remote Management

This step only needs to be completed if you plan to use one of the custom-built extensions for the Keyfactor Universal Orchestrator to manage certificate stores on Windows machines that relies on PowerShell remoting and WinRM. Keyfactor offers many custom-built extensions for the Universal Orchestrator on GitHub:

https://keyfactor.github.io/integrations-catalog/content/orchestrator

Packages that rely on PowerShell remoting and WinRM for store management on Windows include:

- IIS Certificate Store Manager
- Remote File Certificate Store Management (Java Keystores, PKCS12 files, PEM files, DER files, IBM Key Database files)

Permissions

On each target machine where you wish to manage certificate stores with the Universal Orchestrator, you need to grant the Active Directory or local service account the orchestrator is using to authenticate to the server sufficient permissions to read the directories where the certificate stores are located (for the Remote File extension) or local machine certificate store (for the IIS extension) and, if you plan to deploy certificates to it using Keyfactor Command and bind certificates to IIS, write to the directories (Remote File) and local machine store (IIS) and appropriate permissions to bind certificates in IIS. For the Remote File extension, granting read/write permissions on the given directories may be sufficient. For IIS, the Universal Orchestrator service account needs to be added to the local administrators group on each target machine.

PowerShell Remoting

The orchestrator uses PowerShell remoting to deliver certificates to targets and bind certificates to IIS web sites. This includes certificates delivered directly from the PFX enrollment option of the Keyfactor Command Management Portal or Keyfactor API to targets. If you wish to use any of these features, you will need to make sure that each target machine on which you want to use one of these features is running at least PowerShell version 3 and that PowerShell remoting has been enabled.

To check the PowerShell version on a given machine, open a PowerShell window, run the following command, and check the output CLRVersion:

```
$PSVersionTable
```

PowerShell version 3 is available for download from Microsoft.

To enable PowerShell remoting:

1. On the target machine, open a PowerShell window using the "Run as administrator" option.
2. On the target machine, run the following command to enable PowerShell remoting:

   ```
   Enable-PSRemoting
   ```

   Respond Yes to all the question prompts (or A for all).
3. On the target machine it may be necessary to run the following command to enable execution of unsigned local PowerShell scripts for some operating systems (e.g. Windows Server 2008 R2):

   ```
   Set-ExecutionPolicy RemoteSigned
   ```

4. To test the PowerShell remoting, on the Universal Orchestrator server, open a PowerShell window and run the following command (where TARGET_MACHINE is the FQDN of the target machine you wish to manage with the orchestrator):

   ```
   Enter-PSSession –ComputerName TARGET_MACHINE
   ```

   Use the actual hostname of the target machine rather than a DNS alias (either A or CNAME records) when running this test. This is necessary because PowerShell remoting relies on Kerberos authentication, which requires that the target machine has a service principal name (SPN) in the HTTP/ format assigned to the target's machine account. This will be present by default (as part of the HOST/ format record) as long as the HTTP/ format SPN has not been manually assigned elsewhere. Using an alias gets into complexities of setting up appropriate SPNs and assuring that there are not duplicate SPNs in the environment.

   You should be connected to the target machine and be able to execute PowerShell commands on the target machine.

WinRM and Firewall Port Considerations

When you add a certificate store in Keyfactor Command using an extension from Keyfactor's GitHub that relies on WinRM, you are given the option to choose whether to secure the channel to the target hosting the certificate store with SSL. If you select True, Microsoft Windows Remote Management (WinRM) on the target needs to be running on HTTPS and to have been configured with a certificate for WinRM. If you select False, WinRM on the target needs to be running on HTTP. By default, WinRM HTTP uses port 5985 and WinRM HTTPS uses port 5986. WinRM HTTPS is not enabled out-of-the box.

Make sure that any firewalls between the Universal Orchestrator, Keyfactor Command, and the remote target allow communications over port TCP 5985 or 5986, depending on your SSL selection,

or the alternate port you've configured for WinRM on the target if you're not using the default WinRM port(s).

You can use the Test-WSMan and Test-netConnection PowerShell cmdlets on the Universal Orchestrator to validate that communication can occur between the Universal Orchestrator and the remote target in the manner you are intending to configure it (SSL or not SSL). For example, for SSL using the default port (where websrvr38.keyexample.com is your remote target):

```
Test-netConnection -ComputerName "websrvr38.keyexample.com" port 5986
```

Output from this command should look something like this if the connection completes successfully:

```
ComputerName     : websrvr38.keyexample.com
RemoteAddress    : 192.168.216.38
RemotePort       : 5986
InterfaceAlias   : Ethernet0
SourceAddress    : 192.168.216.42
TcpTestSucceeded : True
```

And:

```
Test-WSMan -ComputerName websrvr38.keyexample.com -UseSSL
```

Output from this command should look something like this if the connection completes successfully:

```
wsmid           : http://schemas.dmtf.org/wbem/wsman/identity/1/wsmanidentity.xsd
ProtocolVersion : http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd
ProductVendor   : Microsoft Corporation
ProductVersion  : OS: 0.0.0 SP: 0.0 Stack: 3.0
```

On the remote target, you can use the following WinRM command to check the configuration of WinRM , whether it has been configured to support HTTPS, whether it has a certificate configured for HTTPS, and the ports in use:

```
winrm enumerate winrm/config/listener
```

Output from this command should look something like this if both HTTP and HTTPS are configured for WinRM (notice the port for HTTPS and the certificate thumbprint indicating a certificate has been configured for WinRM on HTTPS):

```
Listener
   Address = *
```

```
    Transport = HTTP
    Port = 5985
    Hostname
    Enabled = true
    URLPrefix = wsman
    CertificateThumbprint
    ListeningOn = 192.168.216.42, 127.0.0.1, ::1, fe80::21e1:ab7e:9c35:5550%3

Listener
    Address = *
    Transport = HTTPS
    Port = 5986
    Hostname = websrvr42.keyexample.com
    Enabled = true
    URLPrefix = wsman
    CertificateThumbprint = 79ee047d673da83cea87ba779761b0ec2b9217f8
    ListeningOn = 192.168.216.42, 127.0.0.1, ::1, fe80::21e1:ab7e:9c35:5550%3
```

For troubleshooting help, see Remote Management Helpful Tools on page 3020. For more information about configuring WinRM for HTTPS, see:

https://learn.microsoft.com/en-us/troubleshoot/windows-client/system-management-components/configure-winrm-for-https

5.2.5.2 Configure the Universal Orchestrator for Remote CA Management

If you've opted to enable the remote CA management functionality for the Keyfactor Universal Orchestrator, further configuration is needed on the orchestrator to configure the CA(s) that the orchestrator will manage.

To configure CAs for the orchestrator:

1. On the orchestrator, open a text editor (e.g. Notepad) using the "Run as administrator" option.
2. In the text editor, browse to open the *extensionoptions.json* file for the Universal Orchestrator. The file is located in the configuration directory within the install directory, which is the following directory by default:

   ```
   C:\Program Files\Keyfactor\Keyfactor Orchestrator\configuration
   ```

3. In the extensionoptions.json file, locate the CertificateAuthority section.

```
{
  "CertificateAuthority": {
    "BatchSize": 10000,
    "CacheHours": 3,
    "RecordCountLimit": 5000,
    "MaxErrorCount": 5,
    "AdditionalCertificateAuthoritiesAllowed": false,
    "CertificateAuthorities": [
      {
        "Forest": "keyother.com",
        "Hostname": "corpca01.keyother.com",
        "LogicalName": "KeyIssuing01"
      },
      {
        "Forest": "keyother.com",
        "Hostname": "corpca02.keyother.com",
        "LogicalName": "KeyIssuing02"
      }
    ]
  },
```


Figure 574: CA Configuration Settings

4. Either set the *AdditionalCertificateAuthoritiesAllowed* value to **true** or populate the *CertificateAuthorities* section with your CA information (see Table 855: Remote CA Configuration Parameters).

5. Save the file.

6. Restart the orchestrator service (see Start the Universal Orchestrator Service on page 2953).

Table 855: Remote CA Configuration Parameters

Parameter	Description
BatchSize	An integer that specifies the number of certificate cache records to read from the Keyfactor Command in each data retrieval batch. The default is 10,000. **Tip:** Certificate cache information from Keyfactor Command is retrieved from and stored on the orchestrator to allow the orchestrator to calculate which records represent changes and return only those to Keyfactor Command on requests from Keyfactor Command for CA synchronization.
CacheHours	An integer that specifies the number of hours for which to cache certificate information from Keyfactor Command on the orchestrator before clearing it. The default is 3.
RecordCountLimit	An integer that specifies the number of records to read from the CA(s) in each synchronization batch. The default is 5,000.
MaxErrorCount	An integer that specifies the number of times an attempt

Parameter	Description
	should be made to read records from the CA before the synchronization job ends with a failure. The default is 5.
AdditionalCertificateAuthoritiesAllowed	A Boolean that sets whether any CAs available to the orchestrator (to which the orchestrator has network access and sufficient permissions) should be considered as managed (*True*) or whether only those CAs specifically listed in the *CertificateAuthorities* parameter should be considered as managed (*False*). If you set this value to *True*, you do not need to populate the *CertificateAuthorities* value.
CertificateAuthorities	An array of the certificate authorities that should be considered managed by the orchestrator. The certificate authority information includes: **Parameter** / **Description** Forest — The name of the Active Directory forest in which the CA resides. Hostname — The fully qualified domain name of the CA. LogicalName — The logical name of the CA.

5.2.5.3 Installing Custom-Built Extensions

Keyfactor offers many custom-built extensions for the Keyfactor Universal Orchestrator on GitHub:

https://keyfactor.github.io/integrations-catalog/content/orchestrator

Some packages that may be of special interest to long-term users of Keyfactor Command are:

- AWS Certificate Store Manager
- Citrix NetScaler Certificate Store Manager
- F5 Certificate Store Manager
- IIS Certificate Store Manager
- Remote File Certificate Store Management (Java Keystores, PKCS12 files, PEM files, DER files, IBM Key Database files)

Tip: Unlike the Keyfactor Java Agent, which must be installed directly on each machine holding Java keystores to be managed, the Keyfactor Universal Orchestrator with the Remote File extension is a centralized orchestrator, which is installed on just a single machine (or handful of machines) and then reaches out via remote management to each machine holding Java Keystores to be managed. For Java keystores on Windows servers, it uses PowerShell remoting and WinRM, typically over HTTPS, for this (see Configure Windows Targets for Remote Management on page 2935). For Java keystores on Linux servers, it uses either SCP or SFTP (configurable on a per-orchestrator basis and can be configured to try both). Large-scale deployment of Java keystore management (or any of the other formats supported by this extension) involves enabling PowerShell remoting and WinRM with HTTPS on Windows targets (and a local login user if the servers aren't domain joined) or enabling SCP or SFTP and creating a login user for the orchestrator on Linux targets.

Note: For information about installing PAM extensions for the Universal Orchestrator, see Installing Custom PAM Provider Extensions on page 743.

To find a package on GitHub:

1. Visit one of the links above to find your desired package, and click either **Github Repository** or **View source on GitHub** to go to the package page on GitHub.

Figure 575: View Packages as Part of a List

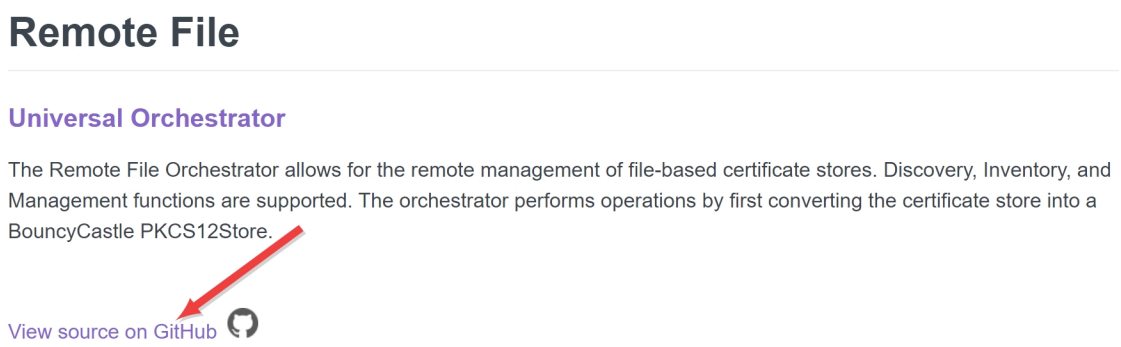


Figure 576: View Packages on Individual Pages

2. On the GitHub page, on the right-hand side, click the link for the **Latest** version.

Figure 577: Find the Latest Version of the Package

3. On the GitHub version page in the Assets section, click the package name to download the zip file.

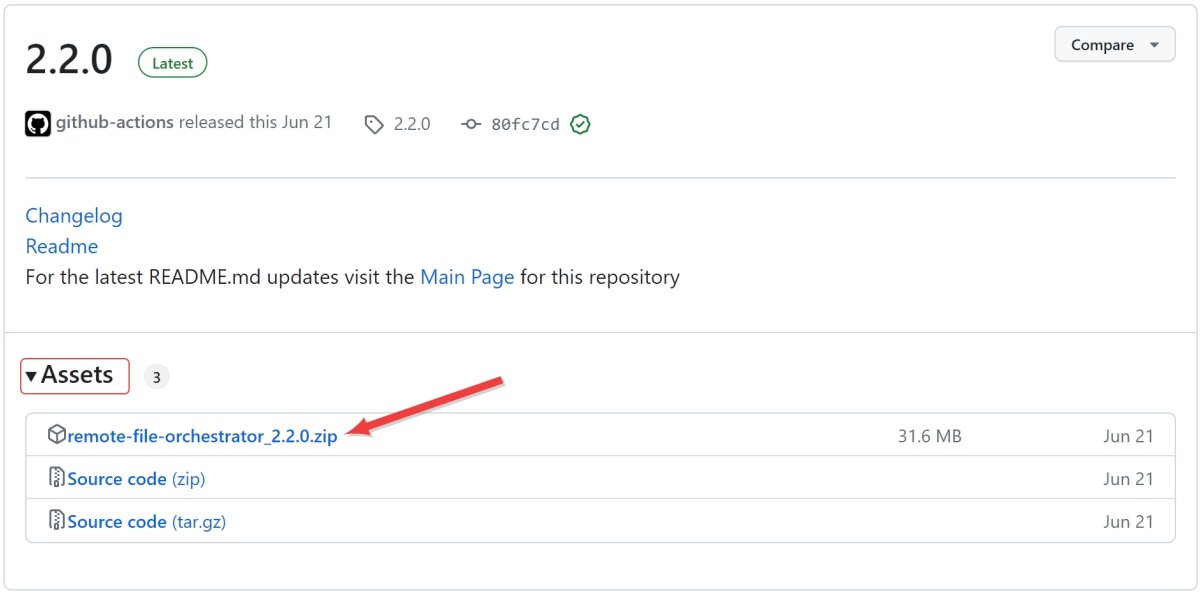


Figure 578: Download the Package Zip File

4. On the main extension GitHub page, review the documentation for the specific extension. Here you will find supported platforms, prerequisites, and extension-specific installation and configuration instructions. The below instructions only cover where to place the extension files on the orchestrator and building custom manifest.json files (changes to which aren't needed for extensions from GitHub unless you are customizing something), but not the details for creation of custom certificate store types for the extension or any other customization specific to a given extension.

Custom-built extensions can also be generated by end users using the Universal Orchestrator NuGet package. Custom-built extensions for certificate store jobs and custom jobs are both installed in the same way.

Once you have your custom-built extension ready, install it as follows:

1. In the Keyfactor Command Management Portal or using the Keyfactor API, add a certificate store type or custom job type for your custom-built extension, if applicable. See Adding or

Editing a Certificate Store Type on page 701 or POST Custom Job Types on page 1600

2. On the Universal Orchestrator server, locate the extensions directory within the install directory. By default, this is:

   Windows: C:\Program Files\Keyfactor\Keyfactor Orchestrator\extensions

   Linux: /opt/keyfactor/orchestrator/extensions

3. Under the extensions directory, create a new directory with an appropriate name for your custom-built extension (e.g. MyExtension). This name is for reference only and does not need to match any names used elsewhere.

4. Place the DLL(s) created for your custom-built extension along with any other supporting files needed for the extension in the new directory.

5. In the directory for your custom-built extension, create a file called manifest.json if one has not been provided with the extension. The manifest.json file must be placed in the same directory as the DLL(s) for your extension.

6. Using a text editor, edit the manifest.json file if needed and configure it appropriately for your application.

   Tip: This step is generally not needed for extensions downloaded from GitHub unless you have opted to make customizations or not use the suggested short name when creating the certificate store type.

   Some things to keep in mind are:

   - The opening and closing lines of the file must match those shown in red here:

```
{
   "extensions": {
      "Keyfactor.Orchestrators.Extensions.IOrchestratorJobExtension":
{
         "Custom.MyJob": {
            "assemblypath": "Keyfactor.Orchestrators.MyJob.dll",
            "TypeFullName": "Keyfactor.Orchestrators.MyJob.MyJobExtension"
         }
      }
   }
}
```

   - Each customized section of the file starts with either a custom job reference (e.g. Custom.MyJob) or a certificate store reference (e.g. CertStores.MyStore.Inventory).

Custom jobs (beginning *Custom*) correspond to custom job types created with the Keyfactor API *POST /JobTypes/Custom* method. For example, a custom job type with a *JobTypeName* of *MyJob* would appear in the file as *Custom.MyJob*.

Certificate store jobs (beginning *CertStores*) correspond to certificate store types created with the Keyfactor API *POST /CertificateStoreType* method (see POST Certificate Store Types on page 1552) or in the Keyfactor Command Management Portal (see Adding or Editing a Certificate Store Type on page 701). For example, a certificate store type with a *Capability* of *MyStore* configured to do inventory, management and discovery, would have three separate sections in the file as *CertStores.MyStore.Inventory*, *CertStores.MyStore.Management*, and *CertStores.MyStore.Discovery*. An inventory section is required.

- The *assemblypath* referenced in each section points to the DLL in the extensions directory that corresponds to that job function. A single manifest file may include many different capabilities if the extension performs more than one type of job (e.g. inventory and management of certificates), such as is shown in the below example.
- The *TypeFullName* referenced in each section corresponds to the name of the type that resides inside of the DLL listed for the assembly path. A single manifest file may include many different capabilities if the extension performs more than one type of job (e.g. inventory and management of certificates), such as is shown in the below example.
- Each section may optionally have a *PreScript* reference, which points to a script file on the orchestrator machine that will run before the main job for the section executes.
    - For orchestrators installed on Windows, these will be PowerShell scripts. No special configuration is needed other than entry of a path to the PowerShell script in the *PreScript* field. The script may be placed anywhere on the orchestrator machine. The orchestrator will need read permissions to the script.
    - For orchestrators installed on Linux, these will be Bash scripts. In order to use a Bash script with the orchestrator, you must first register the Bash script driver in the appsettings.json file. This file is found in the *configuration* directory. Edit the file and add the following below the existing AppSettings configuration section in the file (before the final closing bracket):

      ```
      "extensions": {
         "Keyfactor.Orchestrators.ScriptDrivers.IScriptDriver": {
            "RegisteredScriptDriver": {
               "assemblypath": "Keyfactor.Orchestrators.BashDriver.dll",
               "TypeFullName": "Keyfactor.Orchestrators.ScriptDrivers.BashDriver"
            }
         }
      }
      ```

      After the Bash script driver is registered, you may enter a path to the Bash script in the orchestrator manifest.json file *PreScript* section. The script may be placed

anywhere on the orchestrator machine. The orchestrator will need read permissions to the script.

Tip: If your script fails, this will cause the entire job to fail. You can use this to your advantage if you'd like to fail the job under certain conditions by doing a *Write-Error* on Windows or *exit <error code>* on Linux.

For more information about calling scripts from the orchestrator, contact your Keyfactor representative.

Note: The prescript and postscript functionality of the Keyfactor Universal Orchestrator has been replaced by other functionality in Keyfactor Command such as that provided by Keyfactor Command workflows (see Workflow Definitions on page 230). As a result, prescript and postscript functionality has been deprecated and will be removed from a future release.

- Each section may optionally have a *PostScript* reference, which points to a script file on the orchestrator machine that will run after the main job for the section executes. See the notes for script use under *PreScript*.
- User-defined certificate store jobs support up to four job types—Inventory, Management, Discovery, and Reenrollment. Each one of these job types should have a separate section in the file.

```
{
   "extensions": {
      "Keyfactor.Orchestrators.Extensions.IOrchestratorJobExtension": {
         "CertStores.MyStore.Inventory": {
            "assemblypath": "Keyfactor.Orchestrators.MyStore.dll",
            "TypeFullName": "Keyfactor.Orchestrators.MyStore.MyStoreInventoryJobExtension"
         },
         "CertStores.MyStore.Management": {
            "assemblypath": "Keyfactor.Orchestrators.MyStore.dll",
            "TypeFullName": "Keyfactor.Orches-
trators.MyStore.MyStoreManagementJobExtension",
            "PreScript": "C:\\Program Files\\Keyfactor\\Keyfactor
Orchestrator\\extensions\\MyStoreManagementPreScript.ps1",
            "PostScript": "C:\\Program Files\\Keyfactor\\Keyfactor Orches-
trator\\extensions\\MyStoreManagementPostScript.ps1"
         },
         "CertStores.MyStore.Discovery": {
            "assemblypath": "Keyfactor.Orchestrators.MyStore.dll",
            "TypeFullName": "Keyfactor.Orchestrators.MyStore.MyStoreDiscoveryJobExtension"
         }
```

```
        }
    }
}
```

7. Restart the Universal Orchestrator service (see Start the Universal Orchestrator Service on page 2953).

8. In the Keyfactor Command Management Portal, re-approve the orchestrator. The orchestrator will update to a status of new (if it had been approved previously) upon receiving updated capabilities. See Orchestrator Management on page 496 for information on approving orchestrators.

Contact your Keyfactor representative for more information about custom-built solutions or to obtain access to the NuGet packages required for development of Universal Orchestrator extensions.

5.2.5.4 Configuring Script-Based Certificate Store Jobs

The Keyfactor Universal Orchestrator supports the option to implement custom-built certificate store jobs using one or more scripts (PowerShell or Bash) rather than a full extension (see Installing Custom-Built Extensions on page 2940). To implement custom-built certificate store jobs in this way, you need to create your scripts that will execute the certificate store actions (e.g. inventory, add certificates, remove certificates) and a manifest.json file to reference the jobs and install them on the orchestrator. Optionally, each certificate store action script can call a prescript and/or a postscript to perform actions before or after the main action.

Note: The scripting method of running custom-built certificate store jobs cannot be used to run other types of custom jobs. These are supported only with the use of a custom extension (see Installing Custom-Built Extensions on page 2940). However, both certificate store jobs and custom jobs support the use of prescripts and postscripts (see Orchestrator Job Overview on page 2877).

To configure a set of custom-built certificate store scripts:

1. On the Universal Orchestrator server, locate the scripts directory within the install directory. By default, this is:

   Windows: C:\Program Files\Keyfactor\Keyfactor Orchestrator\Scripts

   Linux: /opt/keyfactor/orchestrator/Scripts

2. Under the scripts directory, create a new directory with an appropriate name for your custom-built certificate store job set (e.g. MyStore). This name matches the name of the job referenced in the manifest.json file.

3. Place the scripts created for your custom-built certificate store job set in the new directory. Supported script file names are:
    - Add (e.g. Add.ps1 or Add.sh)
      A management job to add a certificate to the certificate store.
    - Create (e.g. Create.ps1 or Create.sh)
      A management job to create the certificate store if it does not already exist.
    - Discovery (e.g. Discovery.ps1 or Discovery.sh)
      A discovery job.
    - Inventory (e.g. Inventory.ps1 or Inventory.sh)
      An inventory job.
    - Reenrollment (e.g. Reenrollment.ps1 or Reenrollment.sh)
      A reenrollment job.
    - Remove (e.g. Remove.ps1 or Remove.sh)
      A management job to remove a certificate from the certificate store.
4. In order to use a Bash script with orchestrators installed on Linux, you must first register the Bash script driver in the appsettings.json file. This file is found in the configuration directory. Edit the file and add the following below the existing AppSettings configuration section in the file (before the final closing bracket):

    ```
    "extensions": {
       "Keyfactor.Orchestrators.ScriptDrivers.IScriptDriver": {
          "RegisteredScriptDriver": {
             "assemblypath": "Keyfactor.Orchestrators.BashDriver.dll",
             "TypeFullName": "Keyfactor.Orchestrators.ScriptDrivers.BashDriver"
          }
       }
    }
    ```

5. On the Universal Orchestrator server, locate the JobExtensionDrivers directory within the extensions directory under the install directory. By default, this is:

    Windows: C:\Program Files\Keyfactor\Keyfactor Orchestrator\extensions\JobExtensionDrivers

    Linux: /opt/keyfactor/orchestrator/extensions/JobExtensionDrivers

6. In the JobExtensionDrivers directory, create a file called manifest.json or open the existing one. There should be only one manifest.json file no matter how many script directories you create.
7. Using a text editor, edit the manifest.json file and configure it appropriately for your custom-built certificate store job set. Some things to keep in mind are:

- The opening and closing lines of the file must match those shown in red here:

```
{
   "extensions": {
      "Keyfactor.Orchestrators.Extensions.IOrchestratorJobExtension":
{
         "CertStores.MyStore.Inventory": {
            "assemblypath": "Keyfactor.Orchestrators.JobExtensionDrivers.dll",
            "TypeFullName": "Keyfactor.Orches-
trators.JobExtensionDrivers.InventoryJobExtensionDriver"
         }
      }
   }
}
```

- Each customized section of the file starts with a certificate store reference (e.g. CertStores.MyStore.Inventory). Certificate stores jobs (beginning *CertStores*) correspond to certificate store types created with the Keyfactor API *POST /CertificateStoreType* method (see POST Certificate Store Types on page 1552) or in the Keyfactor Command Management Portal (see Adding or Editing a Certificate Store Type on page 701). For example, a custom certificate store type with a *Capability* of *MyStore* configured to do inventory, management and discovery, would have three separate sections in the file as *CertStores.MyStore.Inventory*, *CertStores.MyStore.Management*, and *CertStores.MyStore.Discovery*. The capability reference (e.g. MyStore) must also match the name you give to the directory where you place your scripts. An inventory section is required.
- The *assemblypath* referenced in each section points to the DLL in the extensions directory of the Job Extensions Driver extension. This built-in extension is used to run custom-built certificate store jobs as scripts. This value will be the same for all entries in the file.
- The *TypeFullName* referenced in each section corresponds to the name of the type that resides inside of the DLL listed for the assembly path—the Job Extensions Driver extension in this case. This value will be the same for all entries in the file.
- Each section may optionally have a *PreScript* reference, which points to an additional script file on the orchestrator machine that will run before the main job for the section executes.

Tip: If either your PreScript or PostScript fails, this will cause the entire job to fail. You can use this to your advantage if you'd like to fail the job under certain conditions by doing a *Write-Error* on Windows or *exit <error code>* on Linux.

Note: The prescript and postscript functionality of the Keyfactor Universal Orchestrator has been replaced by other functionality in Keyfactor Command such as that provided by Keyfactor Command workflows (see Workflow Definitions on page 230). As a result, prescript and postscript functionality has been deprecated and will be removed from a future release.

- Each section may optionally have a *PostScript* reference, which points to an additional script file on the orchestrator machine that will run after the main job for the section executes.
- Custom-built certificate store jobs support up to four job types—Inventory, Management, Discovery, and Reenrollment. Each one of these job types should have a separate section in the file.

```
{
   "extensions": {
      "Keyfactor.Orchestrators.Extensions.IOrchestratorJobExtension": {
         "CertStores.MyStore.Inventory": {
            "assemblypath": "Keyfactor.Orchestrators.JobExtensionDrivers.dll",
            "TypeFullName": "Keyfactor.Orches-
trators.JobExtensionDrivers.InventoryJobExtensionDriver"
         },
         "CertStores.MyStore.Management": {
            "assemblypath": "Keyfactor.Orchestrators.JobExtensionDrivers.dll",
            "TypeFullName": "Keyfactor.Orches-
trators.JobExtensionDrivers.InventoryJobExtensionDriver"
            "PreScript": "C:\\Program Files\\Keyfactor\\Keyfactor
Orchestrator\\scripts\\MyStore\\MyStoreManagementPreScript.ps1",
            "PostScript": "C:\\Program Files\\Keyfactor\\Keyfactor Orches-
trator\\scripts\\MyStore\\MyStoreManagementPostScript.ps1"
         },
         "CertStores.MyStore.Discovery": {
            "assemblypath": "Keyfactor.Orchestrators.JobExtensionDrivers.dll",
            "TypeFullName": "Keyfactor.Orches-
trators.JobExtensionDrivers.InventoryJobExtensionDriver"
         },
         "CertStores.MyStore.Reenrollment": {
            "assemblypath": "Keyfactor.Orchestrators.JobExtensionDrivers.dll",
            "TypeFullName": "Keyfactor.Orches-
trators.JobExtensionDrivers.InventoryJobExtensionDriver"
         }
      }
   }
}
```

8. Restart the Universal Orchestrator service (see Start the Universal Orchestrator Service on page 2953).

9. In the Keyfactor Command Management Portal, re-approve the orchestrator. The orchestrator will update to a status of new (if it had been approved previously) upon receiving updated capabilities. See Orchestrator Management on page 496 for information on approving orchestrators.

Contact your Keyfactor representative for more information about custom solutions or for assistance creating custom scripts.

5.2.5.5 Configure Logging for the Universal Orchestrator

Keyfactor Universal Orchestrator provides extensive logging for visibility and troubleshooting. For more information about troubleshooting, see Troubleshooting on page 3003.

By default, the Keyfactor Universal Orchestrator places its log files in the logs directory under the installed directory, generates logs at the INFO logging level and stores logs for two days before deleting them. If you wish to change these defaults, follow the directions below for your installation type.

Windows Installations

1. On the Windows server where you wish to adjust logging, open a text editor (e.g. Notepad) using the "Run as administrator" option.

2. In the text editor, browse to open the Nlog.config file for the Universal Orchestrator. The file is located in the configuration directory within the install directory, which is the following directory by default:

   C:\Program Files\Keyfactor\Keyfactor Orchestrator\configuration

3. Your Nlog.config file may have a slightly different layout than shown here, but it will contain the five fields highlighted in Figure 579: Universal Orchestrator on Windows NLog.config File. The fields you may wish to edit are:

   - `variable name="logDirectory" value="logs/"`

     The path to the log file location.

     > **Important:** If you choose to change the path for storage of the log files, you will need to create the new directory (e.g. D:\KeyfactorLogs) and grant the Universal Orchestrator service account under which the Keyfactor Orchestrator Service is running full control permissions on this directory.

   - `fileName="${logDirectory}/Log.txt"`

The path and file name of the active orchestrator log file, referencing the logDirectory variable.

- `archiveFileName="${logDirectory}/Log_Archive_{#}.txt"`

The path and file name of previous days' orchestrator log files, referencing the logDirectory variable. The orchestrator rotates log files daily and names the previous files using this naming convention.

- `maxArchiveFiles="2"`

The number of archive files to retain before deletion.

- `name="*" minlevel="Info" writeTo="logfile"`

The level of log detail that should be generated and output to the log file. The default INFO level logs error and some informational data but at a minimal level to avoid generating large log files. For troubleshooting, it may be desirable to set the logging level to DEBUG or TRACE. Available log levels (in order of increasing verbosity) are:

  - OFF—No logging
  - FATAL—Log severe errors that cause early termination
  - ERROR—Log severe errors and other runtime errors or unexpected conditions that may not cause early termination
  - WARN—Log errors and use of deprecated APIs, poor use of APIs, "almost" errors, and other runtime situations that are undesirable or unexpected but not necessarily "wrong"
  - INFO—Log all of the above plus runtime events (startup/shutdown)
  - DEBUG—Log all of the above plus detailed information on the flow through the system
  - TRACE—Maximum log information—this option can generate VERY large log files

```
<variable name="logDirectory" value="logs/" />
<targets>
  <target name="buffered_wrapper" xsi:type="BufferingWrapper" slidingTimeout="true" bufferSize="500" flushTimeout="500">
    <target xsi:type="File" name="logfile" fileName="${logDirectory}/Log.txt" layout="${longdate} ${logger} [${level}] - ${message}"
            archiveFileName="${logDirectory}/Log_Archive_{#}.txt" archiveEvery="Day" archiveNumbering="Rolling" maxArchiveFiles="2" archiveAboveSize="2147483648" />
  </target>
  <target xsi:type="OutputDebugString" name="String" layout="${longdate} ${logger}::${message}" />
  <target xsi:type="Debugger" name="debugger" layout="${longdate} ${logger}::${message}" />
  <target xsi:type="Console" name="console" layout="${logger} ${message}" />
  <target xsi:type="EventLog" name="eventLog" source="Keyfactor Orchestrator"
          eventId="${event-properties:item=eventID}" category="${event-properties:item=categoryID}" layout="${event-properties:item=message}" />
</targets>
<rules>
  <logger name="*-EVENT" minlevel="Info" writeTo="eventLog" final="true" />
  <logger name="*" minlevel="Info" writeTo="console" />
  <logger name="*" minlevel="Info" writeTo="logfile" >
    <filters>
      <when condition="contains('${logger}', 'Quartz') and level &lt; LogLevel.Warn" action="IgnoreFinal" />
      <when condition="starts-with('${logger}', 'Microsoft.Hosting.Lifetime') and level &gt;= LogLevel.Info" action="LogFinal" />
      <when condition="starts-with('${logger}', 'Microsoft.Azure.SignalR') and level &gt;= LogLevel.Debug" action="LogFinal" />
      <when condition="starts-with('${logger}', 'Microsoft.AspNetCore')" action="Ignore" />
      <when condition="starts-with('${logger}', 'Microsoft') and level &lt; LogLevel.Warn" action="Ignore" />
    </filters>
  </logger>
</rules>
```

Figure 579: Universal Orchestrator on Windows NLog.config File

Linux Installations

1. On the orchestrator machine where you wish to adjust logging, open a command shell and change to the directory in which the orchestrator is installed. By default this is /opt/keyfactor/orchestrator.

2. In the command shell in the directory in which the orchestrator is installed, change to the configuration directory.

3. Using a text editor, open the nlog.config file in the configuration directory. Your nlog.config file may have a slightly different layout than shown here, but it will contain the five fields highlighted in the below figure. The fields you may wish to edit are:

    - `variable name="logDirectory" value="logs/"`

      The path to the log file location.

      > **Important:** If you choose to change the path for storage of the log files, you will need to create the new directory (e.g. /opt/kyflogs) and grant the Universal Orchestrator service account under which the keyfactororchestrator-default service is running full control permissions on this directory.

    - `fileName="${logDirectory}/Log.txt"`

      The path and file name of the active orchestrator log file, referencing the logDirectory variable.

    - `archiveFileName="${logDirectory}/Log_Archive_{#}.txt"`

      The path and file name of previous days' orchestrator log files, referencing the logDirectory variable. The orchestrator rotates log files daily and names the previous files using this naming convention.

- `maxArchiveFiles="2"`

  The number of archive files to retain before deletion.

- `name="*" minlevel="Info" writeTo="logfile"`

  The level of log detail that should be generated and output to the log file. The default INFO level logs error and some informational data but at a minimal level to avoid generating large log files. For troubleshooting, it may be desirable to set the logging level to DEBUG or TRACE. Available log levels (in order of increasing verbosity) are:

  - OFF—No logging
  - FATAL—Log severe errors that cause early termination
  - ERROR—Log severe errors and other runtime errors or unexpected conditions that may not cause early termination
  - WARN—Log errors and use of deprecated APIs, poor use of APIs, "almost" errors, and other runtime situations that are undesirable or unexpected but not necessarily "wrong"
  - INFO—Log all of the above plus runtime events (startup/shutdown)
  - DEBUG—Log all of the above plus detailed information on the flow through the system
  - TRACE—Maximum log information—this option can generate VERY large log files

```
<variable name="logDirectory" value="logs/"/>
<targets>
  <target name="buffered_wrapper" xsi:type="BufferingWrapper" slidingTimeout="true" bufferSize="500" flushTimeout="500">
    <target xsi:type="File" name="logfile" fileName="${logDirectory}/Log.txt" layout="${longdate} ${logger} [${level}] - ${message}"
            archiveFileName="${logDirectory}/Log_Archive_{#}.txt" archiveEvery="Day" archiveNumbering="Rolling" maxArchiveFiles="2" archiveAboveSize="2147483648"/>
  </target>
  <target xsi:type="OutputDebugString" name="String" layout="${longdate} ${logger}::${message}"/>
  <target xsi:type="Debugger" name="debugger" layout="${longdate} ${logger}::${message}"/>
  <target xsi:type="Console" name="console" layout="${logger} ${message}"/>
  <target xsi:type="EventLog" name="eventLog" source="Keyfactor Orchestrator"
          eventId="${event-properties:item=eventID}" category="${event-properties:item=categoryID}" layout="${event-properties:item=message}" />
</targets>
<rules>
  <logger name="*-EVENT" minlevel="Info" writeTo="eventLog" final="true" />
  <logger name="*" minlevel="Info" writeTo="console" />
  <logger name="*" minlevel="Info" writeTo="logfile" >
    <filters>
      <when condition="contains('${logger}', 'Quartz') and level &lt; LogLevel.Warn" action="IgnoreFinal" />
      <when condition="starts-with('${logger}', 'Microsoft.Hosting.Lifetime') and level &gt;= LogLevel.Info" action="LogFinal" />
      <when condition="starts-with('${logger}', 'Microsoft.Azure.SignalR') and level &gt;= LogLevel.Debug" action="LogFinal" />
      <when condition="starts-with('${logger}', 'Microsoft.AspNetCore')" action="Ignore" />
      <when condition="starts-with('${logger}', 'Microsoft') and level &lt; LogLevel.Warn" action="Ignore" />
    </filters>
  </logger>
</rules>
```

Figure 580: Universal Orchestrator on Linux NLog.config File

5.2.5.6 Start the Universal Orchestrator Service

The Keyfactor Universal Orchestrator service runs on the orchestrator server and controls orchestrator communications with the Keyfactor Command server. During the configuration process you set the service account under which the orchestrator service will run. The service should start automatically at the conclusion of the installation. To check to see if it's running and start it if necessary, follow the directions below for your installation type.

Windows Installations

The service on Windows is added with a display name of Keyfactor Orchestrator Service (Default) by default.

1. On the Universal Orchestrator server, open the Services MMC.
2. In the Services MMC confirm that the Keyfactor Orchestrator Service is set to a Startup Type of Automatic (if desired). If the service is not running, click the green arrow to start it.

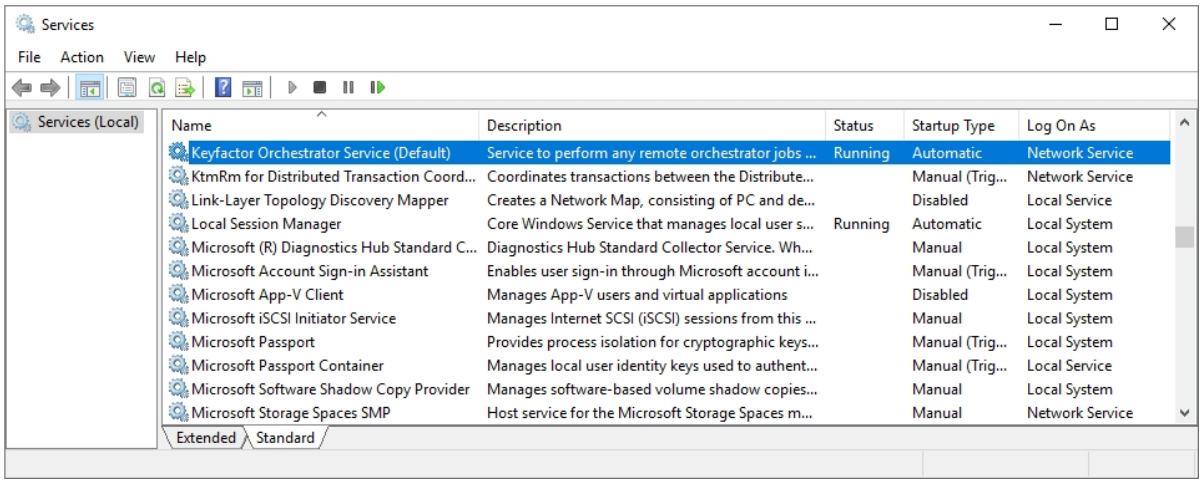


Figure 581: Universal Orchestrator Service

Note: Your service will have a name other than *(Default)* following *Keyfactor Orchestrator Service* if you opted to use the *ServiceSuffix* installation parameter.

Linux Installations

The service on Linux is added as keyfactor-orchestrator-default by default, so when referencing it in startup commands, it should be referenced by this name, including case. For example:

```
systemctl start [stop] [restart] [status] keyfactor-orchestrator-default.service
```

Note: Your service will have a name other than *default* following *keyfactor-orchestrator-* if you opted to use the *service-suffix* installation parameter.

5.2.5.7 Change Service Account Passwords

The process for changing the passwords for the service accounts used by the Keyfactor Universal Orchestrator varies for the different service accounts (see Create Service Accounts for the Universal Orchestrator on page 2884) and based on the type of authentication used for the service account used to connect to Keyfactor Command.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

Universal Orchestrator Service Account

The password for the service account that's used to run the Universal Orchestrator service on the orchestrator server can be changed through standard operating system methods.

On a Linux server, this would be, for example, the command line *passwd* command executed for the service account running the orchestrator service (by default *keyfactor-orchestrator*). So, this command on a Linux server might be:

```
sudo passwd keyfactor-orchestrator
```

On a Windows server, if you've opted to run the Universal Orchestrator service as a custom service account rather than *Network Service*, the password would need to be changed in Active Directory or the local user store and in the Services MMC.

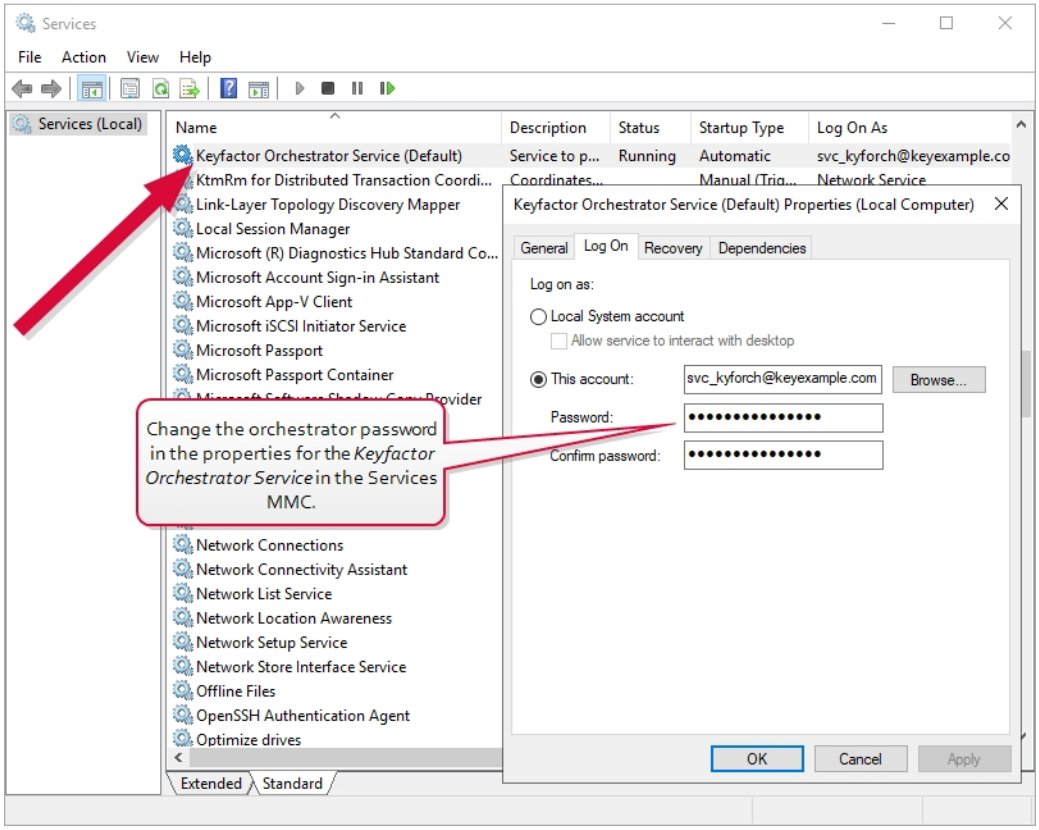


Figure 582: Change Service Account Password in Services MMC

Keyfactor Command Connect Service Account with Basic Authentication

For both Windows and Linux servers, the password change for the service account that's used to make the connection to Keyfactor Command when Basic authentication is used follows this process:

1. Change the password for the service account in Active Directory.

2. On the Windows or Linux server, open a command window. For Windows, this should be a PowerShell window open using the "Run as Administrator" option. Change to the directory in which the orchestrator is installed and locate the change_secrets script. By default, this is:

   ```
   Windows: C:\Program Files\Keyfactor\Keyfactor Orchestrator\change_secrets.ps1
   Linux: /opt/keyfactor/orchestrator/change_secrets.sh
   ```

3. For Linux only, use the chmod command to make the change_secrets.sh script file executable. The file ships in a non-executable state to avoid accidental execution. For example:

   ```
   sudo chmod +x change_secrets.sh
   ```

4. For Windows only, in the PowerShell window, run the following command to populate a variable with the password for the service account:

   ```
   $credKeyfactor = Get-Credential
   ```

   Enter the appropriate username and password when prompted (the service account that the orchestrator uses to connect to Keyfactor Command). Usernames should be given in DOMAIN\username format.

   Or, to avoid being prompted for credentials:

   ```
   $keyfactorUser = "DOMAIN\mykeyfactorconnectusername"
   $keyfactorPassword = "MySecurePassword"
   $secKeyfactorPassword = ConvertTo-SecureString $keyfactorPassword -AsPlainText -
   Force
   $credKeyfactor = New-Object System.Management.Automation.PSCredential
   ($keyfactorUser, $secKeyfactorPassword)
   ```

5. Run the password change script on the Universal Orchestrator server using the following parameters:

   -WebCredential (Windows)

   This is the credential object of the service account that the orchestrator uses to communicate with Keyfactor Command that you created as per Create Service Accounts for the Universal Orchestrator on page 2884. It is provided as a PSCredential object.

   For Basic authentication password change operations, this parameter is **required**.

This parameter cannot be used in conjunction with the *ClientSecret* or *ClientAuthPassword* parameter.

--username (Linux)

The service account that the orchestrator uses to communicate with Keyfactor Command created as per Create Service Accounts for the Universal Orchestrator on page 2884. It may be entered either as username@domain (e.g. svc_kyforch@keyexample.com) or DOMAIN\\username (e.g. KEYEXAMPLE\\svc_kyforch).

For Basic authentication password change operations, this parameter is **required**.

This parameter cannot be used in conjunction with the *client-secret* or *client-auth-password* parameter.

--password (Linux)

The password for the service account that the orchestrator uses to communicate with Keyfactor Command specified with the *username* parameter.

Important: The password for the service account the orchestrator uses to communicate with Keyfactor Command is stored in clear text in the orchestratorsecrets.json file in the configuration directory under the installation directory for the orchestrator. By default, this file is granted read/write permissions for the orchestrator service account running the service on the Linux machine (*keyfactor-orchestrator* by default) and no permissions for any other users. Access to this file should be strictly controlled.

This parameter is **required** if the *username* parameter is specified.

Tip: If you prefer to avoid providing the password at the command line (and storing it in command history), use an input file instead as follows:

a. Create a file that contains just your password. For example:

```
vi my_password_file
```

b. When using the password parameter, reference the file. For example:

```
--password $(cat my_password_file)
```

c. Delete the password file after the install is complete. For example:

```
rm my_password_file
```

-SecretsPath (Windows) or --secrets-path (Linux)

The full path and file name of the or the orchestratorsecrets.json file that stores the secret information. This file is found in the configuration directory under the installation directory for

the Universal Orchestrator, which is by default:

```
Windows: C:\Program Files\Keyfactor\Keyfactor
Orchestrator\configuration\orchestratorsecrets.json
Linux: /opt/keyfactor/orchestrator/configuration/orchestratorsecrets.json
```

The location and file name for this file cannot be changed from the default. The parameter is provided to allow for installations in non-standard locations or multiple locations on the same server.

This parameter is **required**.

Windows example using basic authentication:

```
$keyfactorUser = "KEYXAMPLE\svc_kyforch"
$keyfactorPassword = "MySecurePassword123!"
$secKeyfactorPassword = ConvertTo-SecureString $keyfactorPassword -AsPlainText -Force
$credKeyfactor = New-Object System.Management.Automation.PSCredential ($keyfactorUser, $secKey-
factorPassword)

.\change_secrets.ps1 -WebCredential $credKeyfactor -SecretsPath "C:\Program Files\Keyfactor\Keyfactor
Orchestrator\configuration\orchestratorsecrets.json"

Saved secrets to 'C:\Program Files\Keyfactor\Keyfactor Orches-
trator\configuration\orchestratorsecrets.json'
Restarting service KeyfactorOrchestrator-Default
```

Linux example using basic authentication:

```
vi password_file_new

sudo ./change_secrets.sh --username svc_kyforch@keyexample.com --password $(cat password_file_new) --
secrets-path /opt/keyfactor/orchestrator/configuration/orchestratorsecrets.json

Saving secrets to '/opt/keyfactor/orchestrator/configuration/orchestratorsecrets.json'
Restarting service keyfactor-orchestrator-default
```

Keyfactor Command Connect Service Account with Token Authentication

For both Windows and Linux servers, the secret change for the client that's used to make the connection to Keyfactor Command when token authentication is used follows this process:

1. Change the secret for the client in your identity provider (see Create Service Accounts for the Universal Orchestrator on page 2884).

2. On the Windows or Linux server, open a command window. For Windows, this should be a PowerShell window open using the "Run as Administrator" option. Change to the directory in which the orchestrator is installed and locate the change_secrets script. By default, this is:

   ```
   Windows: C:\Program Files\Keyfactor\Keyfactor Orchestrator\change_secrets.ps1
   Linux: /opt/keyfactor/orchestrator/change_secrets.sh
   ```

3. For Linux only, use the chmod command to make the change_secrets.sh script file executable. The file ships in a non-executable state to avoid accidental execution. For example:

   ```
   sudo chmod +x change_secrets.sh
   ```

4. Run the password change script on the Universal Orchestrator server using the following parameters:

   -ClientSecret (Windows)

   This is the secret of the identity provider client used to authenticate the session with Keyfactor Command (see Create Service Accounts for the Universal Orchestrator on page 2884).

   For token authentication password change operations, this parameter is **required**.

   This parameter cannot be used in conjunction with the *WebCredential* or *ClientAuthPassword* parameter.

   --client-secret (Linux)

   This is the secret of the identity provider client used to authenticate the session with Keyfactor Command (see Create Service Accounts for the Universal Orchestrator on page 2884).

   For token authentication password change operations, this parameter is **required**.

   This parameter cannot be used in conjunction with the *username* and *password* or *client-auth-password* parameters.

   Tip: If you prefer to avoid providing the secret at the command line (and storing it in command history), use an input file instead as follows:

   a. Create a file that contains just your password. For example:

      ```
      vi my_secret_file
      ```

   b. When using the password parameter, reference the file. For example:

      ```
      --client-secret $(cat my_secret_file)
      ```

c. Delete the password file after the install is complete. For example:

```
rm my_secret_file
```

-SecretsPath (Windows) or --secrets-path (Linux)

The full path and file name of the or the orchestratorsecrets.json file that stores the secret information. This file is found in the configuration directory under the installation directory for the Universal Orchestrator, which is by default:

```
Windows: C:\Program Files\Keyfactor\Keyfactor
Orchestrator\configuration\orchestratorsecrets.json
Linux: /opt/keyfactor/orchestrator/configuration/orchestratorsecrets.json
```

The location and file name for this file cannot be changed from the default. The parameter is provided to allow for installations in non-standard locations or multiple locations on the same server.

This parameter is **required**.

Windows example using token authentication:

```
.\change_secrets.ps1 -ClientSecret "aLru2IvZYJh0kFmHa36xs2xTLSpxf4ya" -SecretsPath "C:\Program
Files\Keyfactor\Keyfactor Orchestrator\configuration\orchestratorsecrets.json"

Saved secrets to 'C:\Program Files\Keyfactor\Keyfactor Orches-
trator\configuration\orchestratorsecrets.json'
Restarting service KeyfactorOrchestrator-Default
```

Linux example using token authentication:

```
vi my_new_secret

sudo change_secrets.sh --client-secret $(cat my_new_secret) --secrets-path /opt/key-
factor/orchestrator/configuration/orchestratorsecrets.json

Saving secrets to '/opt/keyfactor/orchestrator/configuration/orchestratorsecrets.json'
Restarting service keyfactor-orchestrator-default

rm my_new_secret
```

Universal Orchestrator Running in a Container

When you're running the Universal Orchestrator in a container, the method for password changes is different and applies only to the Keyfactor Command connect service account. In this scenario, the

approach is to tear down the container and stand up a new one. As long as the new container connects to Keyfactor Command with the same set of capabilities, the same service account username, and the same orchestrator name (either the container hostname or the ORCHESTRATOR_NAME parameter), the orchestrator will continue to operate seamlessly and will not need re-approval.

To change the Keyfactor Command connect service account password for a container:

1. Change the service account's Active Directory password or change the secret for the client in your identity provider (see Create Service Accounts for the Universal Orchestrator on page 2884).

2. Use an appropriate command to tear down the current orchestrator container (being sure you know its configuration). For example:

   ```
   docker compose [-f myconfig.yaml] down
   ```

   Or:

   ```
   kubectl delete -f myconfig.yaml]
   ```

3. Stand up a new container referencing the same set of capabilities, the same service account username, and the same orchestrator name (either the container hostname or the ORCHESTRATOR_NAME parameter) if you want the orchestrator to continue seamlessly without requiring re-approval (see Install the Universal Orchestrator in a Linux Container on page 2922).

5.2.5.8 Register a Client Certificate Renewal Extension

The Keyfactor Universal Orchestrator supports automated renewal of the certificate used for client certificate authentication. It does this using a custom extension point interface on the orchestrator that can be implemented by the end user. When the client certificate used for authentication by the orchestrator is approaching expiration (within 180 days of expiration by default), the extension generates a CSR with a private key and submits the CSR to Keyfactor Command for enrollment. When Keyfactor Command returns the certificate to the orchestrator, it is paired with the private key and installed for use as the client certificate for authentication. The extension both supplies the information for the CSR and holds a dictionary of client parameters (see Build a Client Certificate Renewal Extension on page 2967).

To register a client authentication certificate renewal extension:

1. Create the extension DLL (see Build a Client Certificate Renewal Extension on page 2967).

2. On the Universal Orchestrator server, locate the extensions folder under the install directory for the orchestrator. By default, this is:

   ```
   Windows: C:\Program Files\Keyfactor\Keyfactor Orchestrator\extensions
   Linux: /opt/keyfactor/orchestrator/extensions
   ```

3. Under the extensions directory, create a new directory for your extension (e.g. CertRotation).
4. Place your DLL in the new CertRotation directory.
5. Create a manifest.json file in the CertRotation directory with the following contents:

```
{
   "extensions": {
      "Keyfactor.Orchestrators.Extensions.IOrchestratorRegistrationUpdater": {
         "RegisteredRegistrationUpdater": {
            "assemblypath": "RegistrationUpdater.dll",
            "TypeFullName": "Custom.Registration.Updaters.CustomRegistrationUpdater",
            "config": {
               "DnsSan": "orchestrator_name.keyexample.com",
               "Subject": "CN=Client Certificate Authentication",
               "DataCenter": "WestCoast",
               "ForceRenewal": "False"
            }
         }
      }
   }
}
```

   Only the values shown in red above should be modified from what is shown in this example:

   - The *assemblypath* is the name of your DLL.
   - The example Custom.Registration.Updaters portion of the *TypeFullName* must match the *namespace* in your code. The example CustomRegistrationUpdater portion of the *TypeFullName* must match the *class* in your code.
   - The config section is only needed if you wish to pass configuration values such as a standard DNS SAN or certificate subject into the extension. Those shown here are examples that match the sample code (see Build a Client Certificate Renewal Extension on page 2967).

The certificate renewal process occurs as follows:

1. When each registration or session renewal of the orchestrator service occurs, the orchestrator, in conjunction with underlying Keyfactor Command functionality, checks the expiration date of the client authentication certificate and compares that with the defined client certificate warning period (180 days) and expiry period (30 days) in Keyfactor Command to determine whether a new certificate is needed.

   Note: If the certificate is in the warning period, operations will continue while a new certificate is requested. If the certificate is in the expiry period or already expired, the orchestrator will not be allowed to register a new session when the existing session expires or

the orchestrator service is restarted.
Orchestrator log messages indicating that a certificate is in the warning period look similar to the following:

```
2021-09-17 12:45:59.7927 Keyfactor.Orchestrators.JobEngine.SessionClient [Warn] - Remote
CMS call 'https://keyfactor.keyexample.com/KeyfactorAgents/Session/Register' returned:
Agent certificate is approaching expiration and should be renewed. (A0100007)
```

Orchestrator log messages indicating that a certificate is in the expiry period look similar to the following:

```
2021-09-09 17:27:37.5367 Keyfactor.Orchestrators.JobEngine.SessionClient [Error] - Remote
CMS call 'https://keyfactor.keyexample.com/KeyfactorAgents/Session/Register' returned:
Agent certificate is approaching expiration and must be renewed. (A0100008)
2021-09-09 17:27:37.5642 Keyfactor.Orchestrators.JobEngine.SessionJobExecutor [Error] -
Error in SessionManager: Unable to register session.
   at Keyfactor.Orchestrators.JobEngine.SessionClient.RegisterAsync(IEnumerable`1 capab-
ilities, CancellationToken cancellationToken)
   at Keyfactor.Orchestrators.JobEngine.SessionJobExecutor.Execute(IJobExecutionContext
context)

Error: A0100008
Agent certificate is approaching expiration and must be renewed.
   at Keyfactor.Orchestrators.JobEngine.SessionClient.RegisterAsync(IEnumerable`1 capab-
ilities, CancellationToken cancellationToken)
```

Tip: The length of the warning period and expiry period are defined in Keyfactor Command and are not user-configurable values. Contact support@keyfactor.com if you need to modify these values.

Tip: The orchestrator can be forced into the warning or expiry state before it reaches these based on certificate lifetime using the *POST /Agents/SetAuthCertificateReenrollment* method in the Keyfactor API or the *Request Renewal* button on the Orchestrator Management page of the Keyfactor Command Management Portal. A status of *Request* (1) is the equivalent of the warning period and a status of *Require* (2) is the equivalent of the expiry period.

2. When either the warning period or expiry period is identified, the Keyfactor Universal Orchestrator will pass a value of *true* to the GetCSRInfo method (*newOrchestratorCertRequestedByPlatform* in the sample extension—see Build a Client Certificate Renewal Extension on page 2967). The extension generates a private key and a CSR using CSR

information (e.g. subject, key size) provided or generated by the extension (depending on the extension design), returns the CSR to the orchestrator, which submits it to Keyfactor Command for certificate enrollment.

If the certificate is within the warning period but not within the expiry period, orchestrator activity will be allowed to continue as usual. If the certificate is within the expiry period, a new session will not be granted when the orchestrator requests a new session and orchestrator activity will not be allowed to continue until the orchestrator acquires a new certificate. If the certificate has expired, the certificate rotation cannot take place since the orchestrator cannot authenticate to Keyfactor Command to complete the rotation.

Tip: If a certificate has expired or some other certificate problem is causing the orchestrator not to be able to acquire a session, the orchestrator can be reset using either the Reset button on the Orchestrator Management page in the Keyfactor Command Management Portal or the *POST /Agents/{id}/Reset* method in the Keyfactor API. This removes the certificate history and allows the orchestrator to register for a session with the certificate currently configured in the appsettings.json file under the configuration directory. You will need to re-approve the orchestrator if you reset it.

3. In Keyfactor Command, a certificate is issued based on:

    - The OrchestratorConstants.CertificateAttributes.CERTIFICATE_AUTHORITY and OrchestratorConstants.CertificateAttributes.CERTIFICATE_TEMPLATE values defined in the custom registration handler enroll function.
    - If no values are supplied in the custom registration handler enroll function, the certificate authority and template defined by the *Certificate Authority For Submitted CSRs* and *Template For Submitted CSRs* application settings in the Keyfactor Command Management Portal.

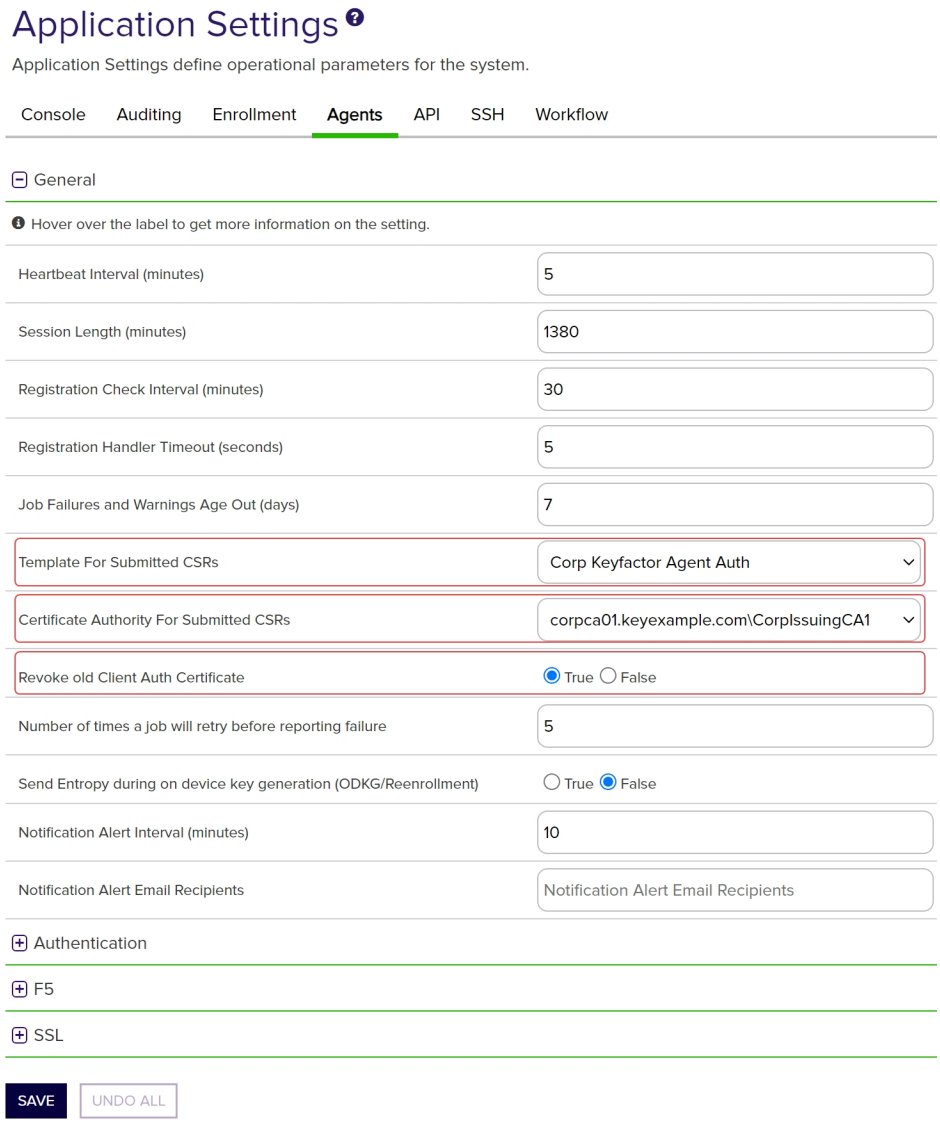


Figure 583: Application Settings for Client Certificate Renewal

4. Once the certificate is issued, it is returned to the orchestrator and married with the private key. If certificate authentication is configured using a certificate stored in the local computer or Universal Orchestrator service account user's personal store (Windows only), the orchestrator updates the *appsettings.json* file with the thumbprint of the new certificate. The thumbprint is stored in the *AuthCertThumbprint* value in the *appsettings.json* file (see Change Service Account Passwords on page 2954). If certificate authentication is configured using a PKCS12 file stored in the file system, a PKCS12 file is generated and replaces the original PKCS12 file. The randomly generated password for the PKCS12 file is updated in the *orchestratorsecrets.json* file.

Note: If certificate authentication is configured using a certificate stored in the local computer personal store on Windows, when the new certificate is generated, it will be placed in the service account user's personal store, not the local computer personal store. This is true if the service is running as a domain account and if the service is running as the default *Network Service*.

5. With the orchestrator's next session registration or heartbeat, it will begin using the new certificate.

Tip: If you manually configured the orchestrator to renew the certificate using the *POST /Agents/SetAuthCertificateReenrollment* method in the Keyfactor API or the *Request Renewal* button on the Orchestrator Management page, that flag will be cleared when the orchestrator successfully registers for a session or completes a heartbeat using the new certificate.
Once a new session is established with the new certificate, the stored legacy thumbprint for the replaced certificate will be removed from the database. This occurs whether or not you opt to automatically revoke the old certificate and occurs before the certificate revocation takes place. The legacy thumbprint is not cleared on a heartbeat with the new certificate.

6. If you've opted to enable the *Revoke old Client Auth Certificate* application setting (see Figure 583: Application Settings for Client Certificate Renewal) in Keyfactor Command, the previous certificate for the orchestrator will be revoked automatically by Keyfactor Command once the orchestrator has made a successful registered for a new session with the new certificate.

Important: The service account under which the Universal Orchestrator is operating must have permissions to enroll for certificates from the CA and have at least the *Enroll CSR* role permission for *Certificate Enrollment* and the *Read* role permission for *Certificates* in Keyfactor Command. If you've opted to enable automated revocation of the old certificate, the service account must also have permissions to revoke certificates on the CA and have at least the *Revoke* role permission for *Certificates* in Keyfactor Command.

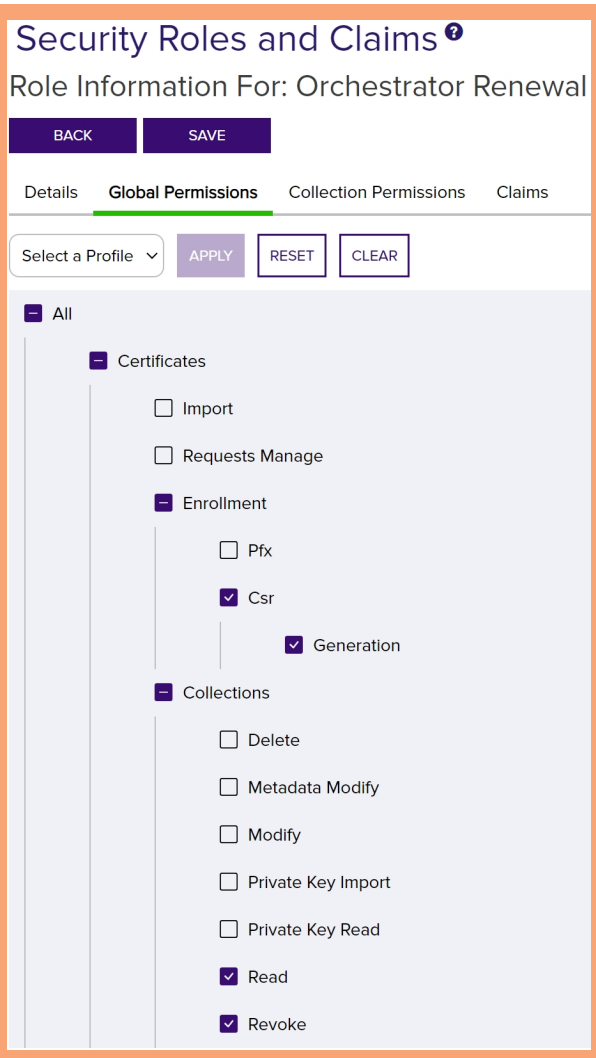


Figure 584: Keyfactor Command Permissions Required for Automatic Renewal and Revocation of Client Authentication Certificates

Build a Client Certificate Renewal Extension

The functionality to renew the certificate used by the Keyfactor Universal Orchestrator for authentication is available via an extension point interface provided by Keyfactor. To implement a custom extension, you will need to obtain the *Keyfactor.Orchestrators.IOrchestratorRegistrationUpdater* nuget package from Keyfactor. Contact your Client Success Manager or support@keyfactor.com for assistance with that.

To build a client certificate renewal extension:

1. Create a project for the extension in your favorite integrated development environment (e.g. Visual Studio).

2. Import the *Keyfactor.Orchestrators.IOrchestratorRegistrationUpdater* nuget package into the project.
3. Consult the sample code to help you design your extension. A sample extension for the client authentication registration updater interface is provided on the Keyfactor GitHub:

   ```
   https://keyfactor.github.io/
   ```

4. Build an assembly file (DLL file) containing the extension.
5. Follow the instructions for registering the extension (see Register a Client Certificate Renewal Extension on page 2961).

5.3 Java Agent

The Keyfactor Java Agent allows organizations to run discovery jobs to locate Java keystores on Windows and Linux systems and PEM certificate stores on Linux systems, inventory the certificates found in them, and push new certificates out to them.

The system requirements for the Java Agent on Windows are:

- 64-bit versions of Windows 8.1, Windows 10, and Windows Server 2019
- 64-bit versions of Oracle Java or OpenJDK 8, 11 or 13
- The WiX Toolset (http://wixtoolset.org/) for users wishing to build an MSI

> **Note:** The path to the WiX executables needs to be added to the System PATH (e.g. C:\Program Files (x86)\WiX Toolset v3.11\bin) to support this.

The system requirements for the Java Agent on Linux are:

- Red Hat 6 or greater, CentOS 6 or greater, or Ubuntu 14 or greater
- 64-bit versions of Oracle Java or OpenJDK 8, 11 or 13
- JSVC on SysV style (init.d) systems

> **Important:** The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:
>
> https://github.com/Keyfactor/remote-file-orchestrator
>
> For more information, see Installing Custom-Built Extensions on page 2940.

5.3.1 Preparing for the Java Agent

This section describes the steps that need to be taken prior to a Java Agent installation to install the prerequisites, create the required supporting components, and gather the necessary information to

complete the Java Agent installation and configuration process.

5.3.1.1 Create Service Accounts for the Java Agent

The Java Agent makes use of up to two service accounts to allow it to communicate with the Keyfactor Command server. These two service accounts work together to transfer information from the Java Agent to the Keyfactor Command server. The two service accounts can be thought of as players on two sides of a fence, with the service account for the Java Agent lobbing information over the fence to the service account on the Keyfactor Command server side to catch and hand to the Keyfactor Command server:

- Java Agent Side
  On the Java Agent side of the fence, you may use either a local account or an Active Directory service account.

  ##### Windows

  For domain-joined Windows machines, an Active Directory service account is typically used. For non-domain-joined Windows machines, you may use a local account created on the Windows machine as the service account instead of a domain account.

  The service account under which the Keyfactor Java Agent service runs on Windows must be granted permissions to "Log on as a service" through local security policy. This step is generally done automatically as part of the installation scripts, but may need to be completed by hand in certain environments or on certain operating systems. The service account needs sufficient permissions to allow it to discover and inventory Java keystores and PEM certificate stores as applicable (read permissions on the appropriate files and directories) and update the stores if desired (write permissions on the files and directories in which the files are stored).

  Important: During the installation process, you enter the Java agent service identity username and password interactively in a PowerShell window to configure the service account. PowerShell will not support the following characters in the service account password when used in this interface:

  ```
  " $
  ```

  If you need to support these characters in the password, you can re-enter the username and password in the Services MMC after receiving an error in the PowerShell interface.

  ##### Linux

  For the purposes of this documentation it is assumed that Linux machines will be non-domain joined and will use a local account to run the Java Agent.

  For Linux systems, Keyfactor recommends running the service as an account other than root.

- Keyfactor Command Server Side
  On the Keyfactor Command server side of the fence, an Active Directory service account in the primary Keyfactor Command server forest is used. This can be the same service account used

for other Keyfactor Command server services. This service account appears in the Management Portal Orchestrator Management grid as the *Identity* for the Java Agent.

If the Java Agent is installed on a domain-joined machine in the same forest as the Keyfactor Command server, the same Active Directory service account may be used on both sides of the fence.

The service accounts need to be created prior to installation of the Java Agent software, and the person installing the Java Agent software needs to know the domain, username and password of each service account.

Important: Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

5.3.1.2 Create a Group for Java Agent Auto-Registration (Optional)

Keyfactor Command can use an Active Directory group to support auto-registration of Java Agents. Auto-registration is an optional feature that allows you to define the conditions under which a Java Agent can automatically be approved for operation with the Keyfactor Command server without administrator input, if desired. This is useful in environments hosting a large number of agents. There are six Java Agent auto-registration roles that share the same AD group:

Java Keystore Discovery

Auto-register the Java Agent to allow it to run discovery tasks to locate Java keystores.

Java Keystore Inventory Reporting

Auto-register the Java Agent to allow it to inventory certificates in Java keystores.

Java Keystore Management

Auto-register the Java Agent to allow it to manage (add/remove) certificates in Java keystores.

PEM Cert Store Management Jobs

Auto-register the Java Agent to allow it to manage (add/remove) certificates in PEM certificate stores.

PEM Server Configuration Directive Parser

Auto-register the Java Agent to allow it to run discovery tasks to locate PEM certificate stores.

PEM Server Inventory

Auto-register the Java Agent to allow it to inventory certificates in PEM certificate stores.

The same Active Directory group must be used for all roles. All auto-registration settings must be populated if any are to be used even if all features are not planned for use. For example, if you plan to use PEM but not Java keystore functionality, you still need to populate the Java keystore auto-registration settings to enable auto-registration for the Java Agent to function correctly.

Note: If all your agents will be connecting to Keyfactor Command as the same service account, you can directly add that user in the auto-registration configuration and skip using a group, if desired.
Although you can choose to enable auto-registration without user validation, allowing any agent to register regardless of the user account under which the agent is running, user validation with either an Active Directory group or a specific Active Directory user is the more secure option.

5.3.1.3 Configure Certificate Root Trust for the Java Agent

Keyfactor recommends using HTTPS to secure the channel between each Java Agent and the Keyfactor Command server(s). This requires an SSL certificate configured in IIS on the Keyfactor Command server(s). This certificate can either be a publicly-rooted certificate (e.g. from Symantec, Entrust, etc.), or one issued from a private certificate authority (CA). If your Keyfactor Command server is using a publicly rooted certificate, the Java Agent machine may already trust the certificate root for this certificate. However, if you have opted to use an internally-generated certificate, your Java Agent server may not trust this certificate. In order to use HTTPS for communications between the Java Agent and the Keyfactor Command server with a certificate generated from a private CA, you will need to import the certificate chain for the certificate into a Java CA certificate store on the Java Agent server. This can be done automatically as part of the installation process. You will need to have the root certificate available as a PEM-encoded format file when you run the installation script.

5.3.1.4 Create Environment Variables for the Java Agent on Windows

The Keyfactor Java Agent determines the location of the current installed Java version on Windows by checking the Windows system environment variables Path and JAVA_HOME. Depending on how your version of Java was installed, these environment variables may or may not be set.

To check and set the environment variables for the Java Agent install:

1. Identify your Java base directory (e.g. C:\Program Files\Java\jdk-13.0.2). This directory typically contains the versioning and release files. Copy this path to a text file for easy access.

2. Identify the location of the Java virtual machine library (e.g. C:\Program Files\Java\jdk-13.0.2\bin\server\jvm.dll), and copy the path to the text file created in the previous step.

3. Identity the location of the main Java executable (e.g. C:\Program Files\Java\jdk-13.0.2\bin\-java.exe), and copy the path to the text file.

4. As a user with local administrator permissions, use the search function to search for *environment* and select the option to edit the *system environment variables* from the search results.

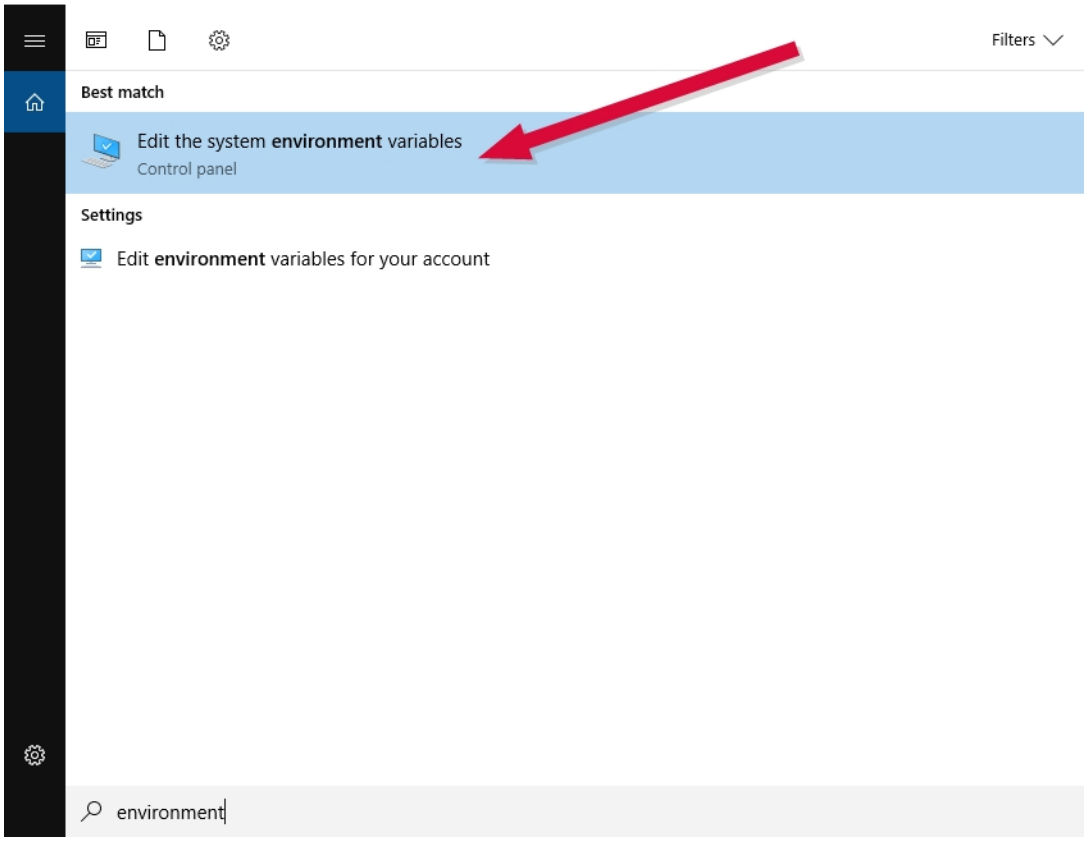


Figure 585: Search for System Environment Variables

5. In the System Properties dialog on the Advanced tab, click *Environment Variables*.

6. In the Environment Variables dialog in the *System variables* section at the bottom, scroll down to locate the *Path* variable, highlight it and click **Edit**.

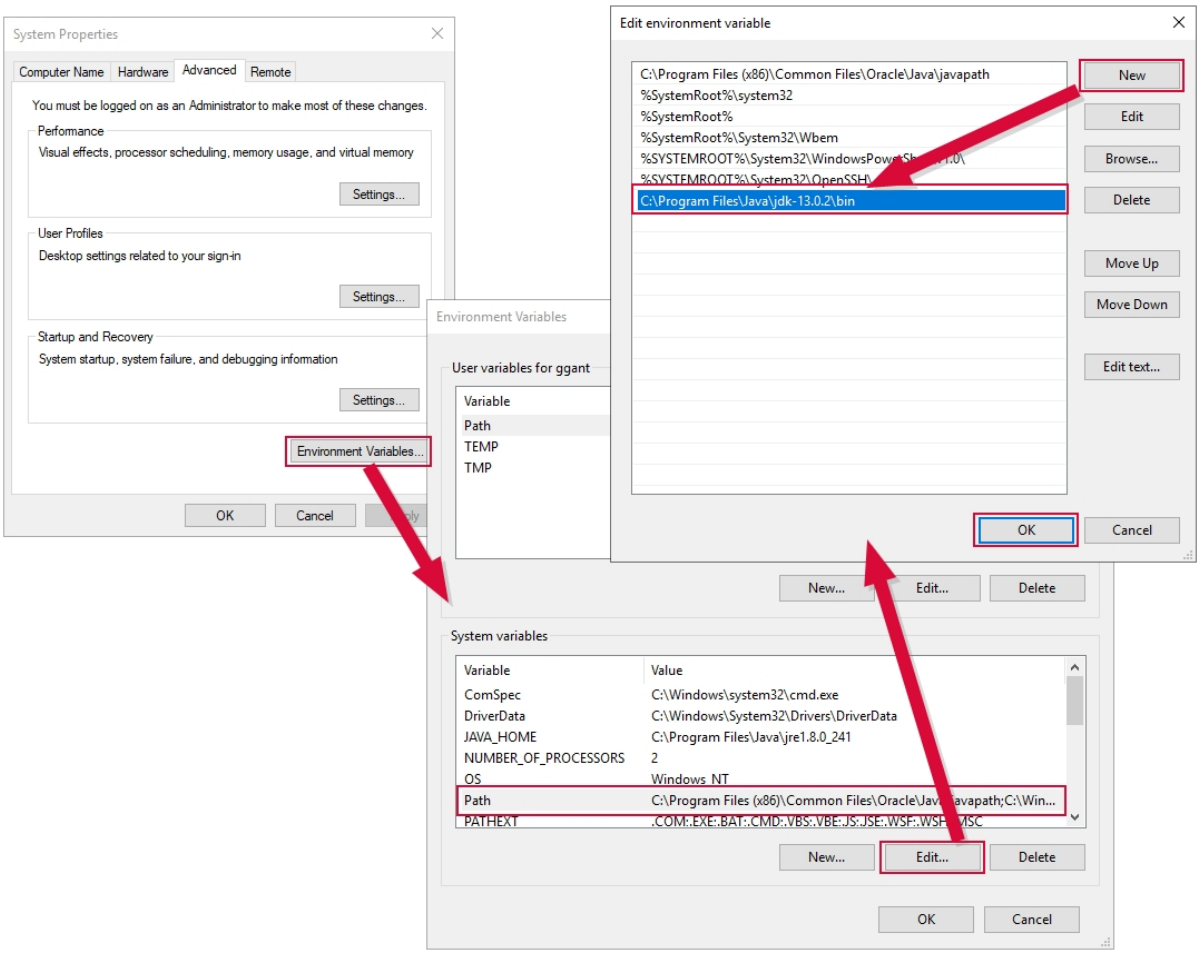


Figure 586: Edit the System Path Environment Variable to Add the Path to Java

7. In the Edit Environment Variable dialog, click **New**. On the newly added line, paste the path to the main Java executable (e.g. C:\Program Files\Java\jdk-13.0.2\bin) that you saved earlier (do not include the java.exe part) and click *OK*.

8. If it doesn't exist already among the *System variables*, create the JAVA_HOME environment variable—click **New** below the *System variables* box and, in the New System Variable dialog, type **JAVA_HOME** in the *Variable name* field and paste the path to the Java base directory in the *Variable value* field. If the field exists already but with a value that is not correct for the version of Java you wish to use, click **Edit** and update the *Variable value* field with the appropriate Java base directory.

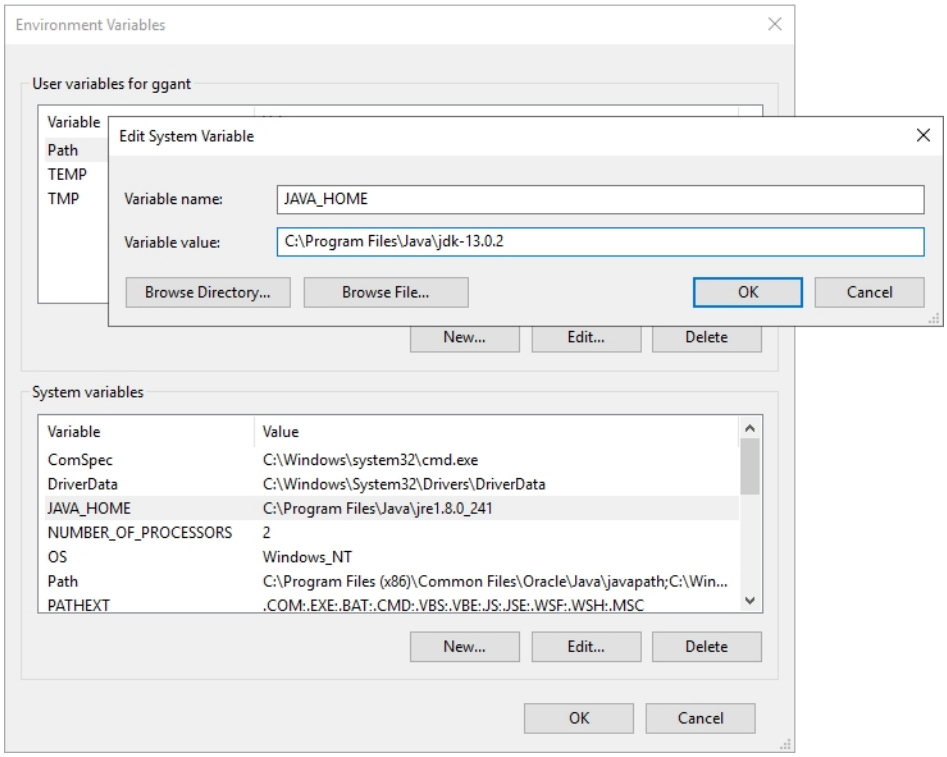


Figure 587: Add JAVA_HOME System Environment Variable

Important: When you run the install.ps1 script, you may be prompted to input the absolute path to the Java Virtual Machine library. From the text file in which you saved paths, take the path to the Java Virtual Machine library (e.g. C:\Program Files\Java\jdk-13.0.2\bin\server-\jvm.dll) and input that string to complete the install.

5.3.2 Install the Java Agent on Windows

The Keyfactor Java Agent installation script offers the option to install the Java agent directly or use the installation script to build an msi package that you can then use to install the Java agent on multiple machines.

Note: If you have a previously installed version of the Keyfactor Java Agent on this server, you need to uninstall it (see Uninstall the Java Agent on page 2990) before installing a new version.

To begin the Java agent installation on Windows, unzip the installation files and place them in a temporary working directory.

1. On the Windows machine on which you wish to install the Java agent or build the package, open a PowerShell window using the “Run as administrator” option and change to the temporary

directory where you placed the installation files.

2. In the PowerShell window, run the cms-java-agent-installer.bat file to begin the installation. You will be prompted to answer several questions:

Username the Java Agent will connect as

This is the service account on the Keyfactor Command server side of the fence you created as per Create Service Accounts for the Java Agent on page 2969. It should be entered in the format DOMAIN\username.

Password for the account that the Java Agent will connect as

This is the password for the service account on the Keyfactor Command server side of the fence.

Hostname or address for the Keyfactor Command Agents server

This is the FQDN or IP address of the Keyfactor Command server running the Keyfactor Command Agent Services role, which is installed as part of the Keyfactor Command Services role. If you installed all the Keyfactor Command server roles together, this is the FQDN or IP address of your Keyfactor Command server.

If you choose to use SSL to connect to the Keyfactor Command server, you'll need to enter a hostname at this prompt that is found in the SSL certificate.

If you're using a non-standard port for IIS on your Keyfactor Command server, enter that here as part of your hostname or IP address (e.g. keyfactor.keyexample.com:444).

Virtual directory for the Keyfactor Command Agents service URL

Press Enter to accept the default of KeyfactorAgents. Only enter an alternate virtual directory if your Keyfactor Command server was configured with an alternate virtual directory for the Keyfactor Command Agents service.

Connect to Keyfactor using SSL?

Press Enter to accept the default of Yes or enter No. The following instructions assume that you answered Yes.

To connect to Keyfactor Command, the Java Agent needs to trust the SSL certificate presented by the Keyfactor Command Agents server

If your Keyfactor Command server is using a publicly rooted certificate, the server most likely already trusts the certificate issuer, and you can press Enter here.

If the certificate on the Keyfactor Command server was internally generated, you will need to enter the full path and file name pointing to a file on the local server containing the PEM-encoded root certificate for the certificate authority chain that issued the certificate (see Configure Certificate Root Trust for the Java Agent on page 2971).

The root certificate will be saved in a Java keystore file called trust.jks located in the Java agent's install directory (C:\Program Files\Keyfactor\Keyfactor Java Agent by default). The default keystore password is *changeit*. Please contact Keyfactor technical support for assistance in changing the default password, if desired.

This question will not appear if you answered no to the question about using SSL.

Verify Keyfactor Command connectivity?

Press Enter to accept the default of "Yes". The Java agent will attempt to connect to the Keyfactor Command server using the credentials provided to confirm that the server name, agents URL, root trust, and provided credentials are valid. Enter "No" to skip this validation if you don't have connectivity to the Keyfactor Command server at the time of installation.

Tip: If the installer terminates after this question without an error or with an error writing the trust.jks file, it can be an indication that the path to the root certificate you provided in the previous question was incorrect in some way (e.g. the path is not valid, the root certificate doesn't match the certificate on the Keyfactor Command server, etc.)

Please specify the installation format

The options at this prompt are "local" or "msi". If you press enter to accept "local", the Java agent will be installed locally. If you enter "msi", the batch file will generate an msi after all the questions have been answered. You can use this to install the Java agent on other Windows systems with the installation questions already answered. The subsequent questions differ depending on the answer given to this question. The following instructions include both local and msi questions. You will not see all of these questions.

If you select "msi", the Java agent will not be installed locally.

Path to the desired directory for installation (Local)

Press Enter to accept the default installation directory of C:\Program Files\Keyfactor\Keyfactor Java Agent or enter an alternate path if desired. This question does not appear when generating an msi.

Local user account the agent should run as \ User account on the target machine that the agent should run as (Local\MSI)

Press Enter to accept the default of the local SYSTEM account for local installs (this is not an option when generating an msi) or enter a specific user account—the service account for the

Java agent side of the fence you created as per Create Service Accounts for the Java Agent on page 2969. Domain user accounts should be entered in the format DOMAIN\username. You do not need to enter the password for this user at this time. The username is entered at this time to allow permissions to be configured appropriately.

Hostname the agent will connect to Keyfactor as (Local)

Press Enter to accept the default of the local machine's hostname as determined by a reverse DNS lookup or, failing that, the value of the local environment variable for the computer name. If desired, you can enter an alternative value to use as the hostname. This is the identifier for the server on which you are installing the Java agent. This identifier can be in the form of a host-name or FQDN, but you can use another unique identifier, if desired. This identifier appears in the Keyfactor Command Management Portal on the orchestrators page. This question does not appear when generating an msi.

Tip: When installing from an msi, you can specify a custom hostname by using the AGENTNAME parameter. In order to use this option, you must install the msi from the command line. For example:

```
msiexec /i C:\temp\cms-java-agent.msi AGENTNAME=jvagnt38.keyexample.com
```

Note: If the agent machine has a non-private address, you will most likely need to use this option.

Directory where the agent logs should be placed (Local)

Press Enter to accept the default log directory of C:\CMS\logs or enter an alternate path if desired. This question does not appear when generating an msi.

Number of log files that should be kept (Local\MSI)

Press Enter to accept the default of 7 log files or enter an alternate number if desired. Older files are automatically deleted once more files than this have been generated.

Maximum size of each log file (Local\MSI)

Press Enter to accept the default log file size of 3 MB or enter an alternate value if desired.

Register AnyAgent components with the Keyfactor Java Agent? (Local)

Press Enter to accept the default value and begin the installation. If you would like to install one or more Any Agent implementations, enter yes. In this case, you'll be presented with a list of custom certificate store types for which to provide an implementation. After choosing each one, you'll need to enter the path to the .jar file that implements the certificate store type. That .jar file will be copied to the installation directory, under the libs folder. You'll need to manually copy any other dependent .jar files to that location as well. Note that this option is only available when

the Java agent is installed locally. This question does not appear when generating an msi.

3. After answering the AnyAgent components question, the installation begins. Review the output to be sure that no errors have occurred and then press any key to return to the PowerShell prompt.

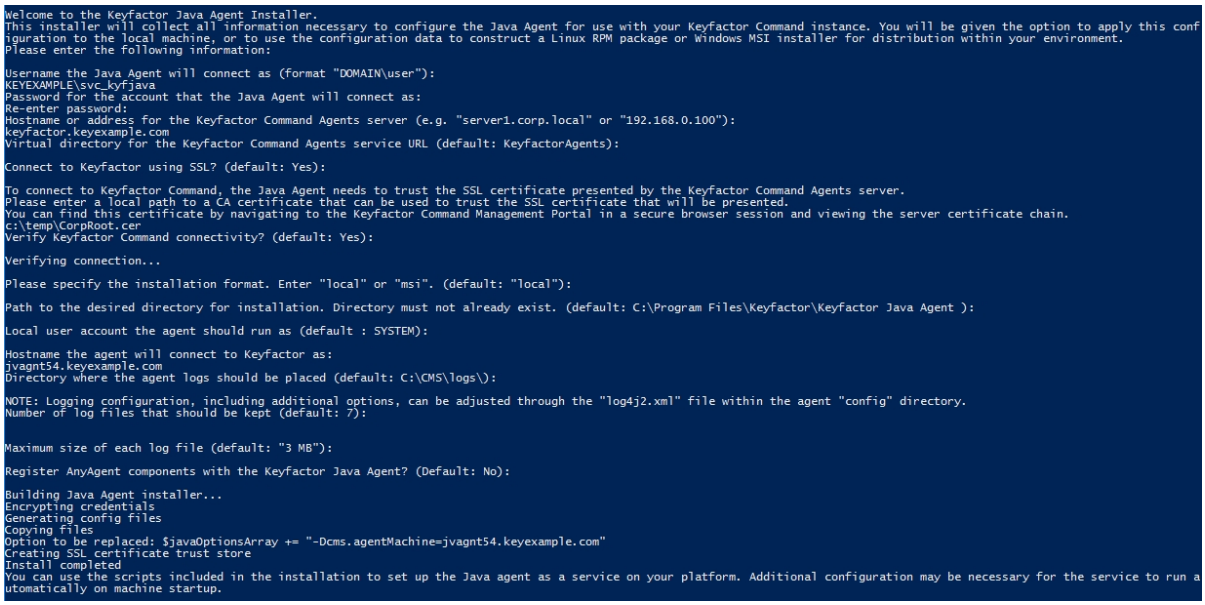


Figure 588: Keyfactor Java Agent Local Installation on Windows

4. In the PowerShell window, change to the install directory within the directory in which you installed the Java agent. If you installed in the default install directory, this path is:

   C:\Program Files\Keyfactor\Keyfactor Java Agent\install

5. In the PowerShell window, run the install.ps1 PowerShell script. Unless you selected SYSTEM as the user the agent should run as, you will be prompted to enter the username (DOMAIN\username format) and password of the account that will run the Keyfactor Java Agent service on the local machine. This is the service account for the Java agent side of the fence you created as per Create Service Accounts for the Java Agent on page 2969. Press Enter without entering any data to run the service under the local system credentials.

   Note: The install.ps1 may fail with an error similar to the following on older versions of Windows:

   ```
   Method invocation failed because [System.Object[]] doesn't contain a method named 'Replace'.
   ```

   If this occurs, you need to manually grant the service account under which the Keyfactor Java Agent service will run the local ”Log on as a service” permission and then run the install.ps1 script again.

Tip: If you choose the “msi” option rather than the “local” option, the MSI file will be generated in the directory in which you executed the batch file.

5.3.3 Install the Java Agent on Linux

The Java Agent installation script offers the option to install the Java Agent directly or use the installation script to build an RPM package that you can then use to install the Java Agent on multiple machines.

Note: If you have a previously installed version of the Keyfactor Java Agent on this server, you need to uninstall it (see Uninstall the Java Agent on page 2990) before installing a new version.

To begin the Java Agent installation on Linux, unzip the installation files and place them in a temporary working directory.

1. On the Linux machine on which you wish to install the Java Agent or build the package, at a command shell change to the temporary directory where you placed the installation files.
2. Use the chmod command to make the cms-java-agent-Installer.sh script executable. The file ships in a non-executable state to avoid accidental execution. For example:

   ```
   sudo chmod +x cms-java-agent-installer.sh
   ```

3. In the command shell, run the cms-java-agent-Installer.sh script *as root* to begin the installation. You will be prompted to answer several questions:

   #### Username the Java Agent will connect as

   This is the service account on the Keyfactor Command server side of the fence you created as per Create Service Accounts for the Java Agent on page 2969. It should be entered in the format DOMAIN\username.

   #### Password for the account that the Java Agent will connect as

   This is the password for the service account on the Keyfactor Command server side of the fence.

   #### Hostname or address for the Keyfactor Command Agents server

   This is the FQDN or IP address of the Keyfactor Command server running the Keyfactor Command Agent Services role, which is installed as part of the Keyfactor Command Services role. If you installed all the Keyfactor Command server roles together, this is the FQDN or IP address of your Keyfactor Command server.

If you choose to use SSL to connect to the Keyfactor Command server, you'll need to enter a hostname at this prompt that is found in the SSL certificate.

If you're using a non-standard port for IIS on your Keyfactor Command server, enter that here as part of your hostname or IP address (e.g. keyfactor.keyexample.com:444).

Virtual directory for the Keyfactor Command Agents service URL

Press Enter to accept the default of KeyfactorAgents. Only enter an alternate virtual directory if your Keyfactor Command server was configured with an alternate virtual directory for the Keyfactor Command Agents service.

Connect to Keyfactor using SSL?

Press Enter to accept the default of Yes or enter No. The following instructions assume that you answered Yes.

To connect to Keyfactor Command, the Java Agent needs to trust the SSL certificate presented by the Keyfactor Command Agents server...

If your Keyfactor Command server is using a publicly rooted certificate, the server most likely already trusts the certificate issuer, and you can press Enter here.

If the certificate on the Keyfactor Command server was internally generated, you will need to enter the full path and file name pointing to a file on the local server containing the PEM-encoded root certificate for the certificate authority chain that issued the certificate (see Configure Certificate Root Trust for the Java Agent on page 2971).

The root certificate will be saved in a Java keystore file called trust.jks located in the Java Agent's install directory (/opt/keyfactor-java-agent by default). The default keystore password is *changeit*. Please contact Keyfactor technical support for assistance in changing the default password, if desired.

This question will not appear if you answered no to the question about using SSL.

Verify Keyfactor Command connectivity?

Press Enter to accept the default of "Yes". The Java Agent will attempt to connect to the Keyfactor Command server using the credentials provided to confirm that the server name, agents URL, root trust, and provided credentials are valid. Enter "No" to skip this validation if you don't have connectivity to the Keyfactor Command server at the time of installation.

Tip: If the installer terminates after this question without an error or with an error writing the trust.jks file, it can be an indication that the path to the root certificate you provided in the previous question was incorrect in some way (e.g. the path is not valid, the root certificate doesn't match the certificate on the Keyfactor Command server, etc.)

Please specify the installation format

The options at this prompt are “local” or “rpm”. If you press enter to accept “local”, the Java Agent will be installed locally. If you enter “rpm”, the script will generate an rpm after all of the questions have been answered. You can use this to install the Java Agent on other Linux systems with the installation questions already answered. The subsequent questions differ depending on the answer given to this question. The following instructions include both local and rpm questions. You will not see all of these questions.

If you select “rpm”, the Java agent will not be installed locally.

Path to the desired directory for installation (Local)

Press Enter to accept the default installation directory of /opt/keyfactor-java-agent or enter an alternate path if desired. This question does not appear when generating an rpm.

Local user account the agent should run as \ User account on the target machine that the agent should run as (Local\RPM)

This is the service account for the Java Agent side of the fence you created as per Create Service Accounts for the Java Agent on page 2969. It should be entered as just the user name. Entry of the password for this service account is not required. The username is entered at this time to allow permissions to be configured appropriately.

Hostname the agent will connect to Keyfactor as (Local)

Press Enter to accept the default of the local machine’s hostname as determined by a reverse DNS lookup or, failing that, the value of the local environment variable for the computer name. If desired, you can enter an alternative value to use as the hostname. This is the identifier for the server on which you are installing the Java agent. This identifier can be in the form of a host-name or FQDN, but you can use another unique identifier, if desired. This identifier appears in the Keyfactor Command Management Portal on the orchestrators page. This question does not appear when generating an rpm.

Full path to the desired buildroot directory for RPM package staging. Directory must not exist. (RPM)

Press Enter to accept the default path of /temp under the current directory or enter an alternate path if desired. This is a temporary location the build process will use while the package is being created. This is not the directory where the final RPM file will be placed. This question does not appear when installing locally.

Note: Ensure the path does not contain spaces. Any space in the java agent path causes issues when building an rpm.

> **Tip:** The RPM file will be generated in a subdirectory (rpmbuild/RPMS) of the home directory of the user running the cms-java-agent-Installer.sh script. If you run the script as root, this will be root's home directory, so you may choose to run the script as a non-root user if you plan to create an RPM.

Path the RPM will install to on the target machine (RPM)

Press Enter to accept the default installation directory of /opt/keyfactor-java-agent or enter an alternate path if desired. This question does not appear when installing locally.

Architecture of the RPM target machine (RPM)

Press Enter to accept the default as determined by the machine on which the RPM is being generated or enter an alternate architecture if desired. A separate RPM needs to be generated with each required machine architecture. This question does not appear when installing locally.

Directory where the agent logs should be placed (Local\RPM)

Press Enter to accept the default log directory of /opt/cms-java-agent/logs or enter an alternate path if desired.

Number of log files that should be kept (Local\RPM)

Press Enter to accept the default of 7 log files or enter an alternate number if desired. Older files are automatically deleted once more files than this have been generated.

Maximum size of each log file (Local\RPM)

Press Enter to accept the default log file size of 3 MB or enter an alternate value if desired.

Register AnyAgent components with the Keyfactor Java Agent? (Local)

Press Enter to accept the default value and begin the installation. If you would like to install one or more Any Agent implementations, enter yes. In this case, you'll be presented with a list of custom certificate store types for which to provide an implementation. After choosing each one, you'll need to enter the path to the .jar file that implements the certificate store type. That .jar file will be copied to the installation directory, under the libs folder. You'll need to manually copy any other dependent .jar files to that location as well. Enter "Done" when you've finished listing agent implementations. Note that this option is only available when the JavaAgent is installed locally.

4. After answering the log file size question, the installation begins. Review the output to be sure that no errors have occurred.

```
Welcome to the Keyfactor Java Agent Installer.
This installer will collect all information necessary to configure the Java Agent for use with your Keyfactor Command instance. You will be given the option to a
pply this configuration to the local machine, or to use the configuration data to construct a Linux RPM package or Windows MSI installer for distribution within
your environment.
Please enter the following information:

Username the Java Agent will connect as (format "DOMAIN\user"):
KEYEXAMPLE\svc_kyfjava
Password for the account that the Java Agent will connect as:
Re-enter password:
Hostname or address for the Keyfactor Command Agents server (e.g. "server1.corp.local" or "192.168.0.100"):
keyfactor.keyexample.com
Virtual directory for the Keyfactor Command Agents service URL (default: KeyfactorAgents):

Connect to Keyfactor using SSL? (default: Yes):

To connect to Keyfactor Command, the Java Agent needs to trust the SSL certificate presented by the Keyfactor Command Agents server.
Please enter a local path to a CA certificate that can be used to trust the SSL certificate that will be presented.
You can find this certificate by navigating to the Keyfactor Command Management Portal in a secure browser session and viewing the server certificate chain.
/tmp/CorpRoot.crt
Verify Keyfactor Command connectivity? (default: Yes):

Verifying connection...

Please specify the installation format. Enter "local" or "rpm". (default: "local"):

Path to the desired directory for installation. Directory must not already exist. (default: /opt/keyfactor-java-agent ):

Local user account the agent should run as:
kyfuser

Hostname the agent will connect to Keyfactor as:
jvagnt162.keyexample.com
Directory where the agent logs should be placed (default: /opt/keyfactor-java-agent/logs/):

NOTE: Logging configuration, including additional options, can be adjusted through the "log4j2.xml" file within the agent "config" directory.
Number of log files that should be kept (default: 7):


Maximum size of each log file (default: "3 MB"):

Register AnyAgent components with the Keyfactor Java Agent? (Default: No):

Building Java Agent installer...
Encrypting credentials
Generating config files
Copying files
Creating SSL certificate trust store
Install completed
You can use the scripts included in the installation to set up the Java agent as a service on your platform. Additional configuration may be necessary for the se
rvice to run automatically on machine startup.
```

Figure 589: Keyfactor Java Agent Local Installation on Linux

5. Keyfactor provides scripts that can be used to configure the Keyfactor Java Agent to start automatically. These can be used on systems using startups based on SysV style (init.d) or systemd. Other startup systems will need to be configured manually. If your machine has neither of these startup systems, you will not be able to use these scripts to configure the Keyfactor Java Agent to start automatically. The appropriate startup script to use depends on whether you are doing a local install or installing from a previously generated RPM file.

   #### Local Install

   a. In the command shell, change to the directory in which you installed the Java Agent. The default install directory is:

      ```
      /opt/keyfactor-java-agent
      ```

   b. Select the appropriate installation script for your startup system. The two available scripts for local installs are:

      ```
      install-init-service.sh
      install-systemd-service.sh
      ```

> **Tip:** The scripts with -with-configured-hostname in their names (e.g. install-systemd-service-with-configured-hostname.sh) are for use with installations from RPM packages and should not be used for local installs.

c. Use the chmod command to make the desired script executable. The file ships in a non-executable state to avoid accidental execution. For example:

```
sudo chmod +x install-systemd-service.sh
```

d. Run the appropriate shell script as root. This will add the keyfactor-java-agent as a service, which you can then stop and start using the standard service stop and start commands. For example:

```
service keyfactor-java-agent restart
systemctl restart keyfactor-java-agent.service
```

Install from RPM

a. Locate the RPM file on the machine on which it was generated and copy it to the machine on which you wish to install the Java agent.

> **Tip:** The RPM file is generated in a subdirectory (rpmbuild/RPMS) of the home directory of the user running the cms-java-agent-Installer.sh script. If you run the script as root, this will be root's home directory.

b. Execute the RPM as root. For example:

```
sudo rpm -ivh keyfactor-java-agent-8.6.0-1.i686.rpm
```

c. In the command shell, change to the directory in which you installed the Java Agent. The default install directory is:

```
/opt/cms-java-agent
```

d. There are four possible installation scripts for installation from RPM packages:

```
install-init-service.sh
install-init-service-with-configured-hostname.sh
install-systemd-service.sh
install-systemd-service-with-configured-hostname.sh
```

Select the appropriate installation script type for your startup system (init or systemd). The versions of the scripts that contain a reference to *with-configured-hostname* in the file name allow you to enter a custom agent name (see Hostname the agent will connect to

Keyfactor as (Local) on page 2981). The versions without this reference will use the system hostname as the agent name.

e. Use the chmod command to make the desired script executable. The file ships in a non-executable state to avoid accidental execution. For example:

```
sudo chmod +x install-systemd-service-with-configured-hostname.sh
```

f. Run the appropriate shell script as root. You will be prompted to answer questions specific to the machine on which the Java agent is being installed—the hostname or other identifier for the machine (see Hostname the agent will connect to Keyfactor as (Local) on page 2981) if you used a *with-configured-hostname* script and the username and password for the service account that will connect the agent to Keyfactor Command (see Username the Java Agent will connect as on page 2979).

g. Change the ownership on the file containing the startup credentials to the local user that the agent will run as. This file is found in the *config* directory under the installed directory and is called *install.creds*. For example:

```
sudo chown kyfuser /opt/cms-java-agent/config/install.creds
```

h. The install shell script adds the keyfactor-java-agent as a service, which you can then stop and start using the standard service stop and start commands. You may need to restart the service after changing the ownership on the credentials file. For example:

```
service keyfactor-java-agent restart
systemctl restart keyfactor-java-agent.service
```

Tip: If desired, you can pass the responses to the questions the installer asks in from a file. For example, for a full install (not working from an RPM file you previously created), create a file that contains values something like this:

```
KEYEXAMPLE\svc_kyfjava
MyVerySecurePassword
MyVerySecurePassword <- The installer requires entry of the password twice
keyfactor.keyexample.com
KeyfactorAgents
Yes
/tmp/CorpRoot.crt
Yes
local
/opt/keyfactor-java-agent
kyfuser
jvagnt162.keyexample.com
```

```
/opt/keyfactor-java-agent/logs
7
"3 MB"
No
```

Note that the values needed in your input file will vary depending on how you answer some of the questions. For example, the first Yes shown above will go in response to the question of whether to use SSL for the connection to Keyfactor Command. If you answer No here, you will not receive the question about needing a root certificate, and so the path to a root certificate shown after this will not correctly match the next question. The script will fail.

Place the file in the same directory as the install script. Then, execute the install script like this:

```
sudo ./cms-java-agent-installer.sh < myinputfile.txt
```

5.3.4 Optional Configuration

Once the installation scripts are complete, the Java Agent should be running and ready to communicate with the Keyfactor Command server.

Important: Java Agent tasks will not run until you complete the Java Agent configuration by making the appropriate configuration changes in the Keyfactor Command Management Portal. See Orchestrators on page 481 in the *Keyfactor Command Reference Guide* for instructions on approving the Java Agent in the Keyfactor Command Management Portal on the *Orchestrators->Auto-Registration* and *Orchestrators->Management* pages, and on configuring certificate stores on the *Certificate Management->Certificate Stores* page (see Adding or Modifying a Certificate Store on page 413 and Certificate Store Discovery on page 437).

5.3.4.1 Configure Logging for the Java Agent

By default, the Java Agent places its log files in the C:\CMS\logs directory on Windows and the /opt/keyfactor-java-agent/logs directory on Linux, generates logs at the *Info* logging level and stores seven 3 MB logs before deleting them (how long this will be will depend on the logging level and the volume of usage the Java Agent is receiving).

If you wish to change these defaults after the installation is complete on Windows:

1. On the Java Agent machine where you wish to adjust logging, open a text editor (e.g. Notepad) using the “Run as administrator” option.

2. In the text editor, browse to open the log4j2.xml file in the config directory under the directory in which you installed the Java Agent. By default, the file is located in the following directory:

   ```
   C:\Program Files\Keyfactor\Keyfactor Java Agent\config
   ```

3. Your log4j2.xml file may have a slightly different layout than shown here, but it will contain the four fields highlighted in the below figure. The fields you may wish to edit are:

   ```
   fileName="C:\CMS\logs\CMS-Java.txt"
   ```

   The path and file name of the active Java Agent log file.

   > **Important:** If you choose to change the path for storage of the log files, you will need to create the new directory (e.g. D:\KeyfactorLogs) and grant the service account under which the Keyfactor Java Agent service is running full control permissions on this directory.

   ```
   size="3 MB"
   ```

   The maximum file size of each log file. After a log file reaches the maximum size, it is rotated to an archive file name and a new log file is generated.

   ```
   max="7"
   ```

   The number of archive files to retain before deletion.

   ```
   level="info"
   ```

   The level of log detail that should be generated. The default info level logs error and some informational data but at a minimal level to avoid generating large log files. For troubleshooting, it may be desirable to set the logging level to debug or trace. Available log levels (in order of increasing verbosity) are:

   - OFF – No logging
   - FATAL – Log severe errors that cause early termination
   - ERROR – Log severe errors and other runtime errors or unexpected conditions that may not cause early termination
   - WARN – Log errors and use of deprecated APIs, poor use of APIs, almost errors, and other runtime situations that are undesirable or unexpected but not necessarily wrong
   - INFO – Log all of the above plus runtime events (startup/shutdown)
   - DEBUG – Log all of the above plus detailed information on the flow through the system
   - TRACE – Maximum log information—this option can generate VERY large log files

Figure 590: Configure Logging for Keyfactor Java Agent on Windows

If you wish to change these defaults after the installation is complete on Linux:

1. On the Java Agent machine where you wish to adjust logging, open a command shell and change to the directory in which the Java Agent is installed. By default this is /opt/keyfactor-java-agent.

2. In the command shell in the directory in which the Java Agent is installed, change to the config directory.

3. Using a text editor, open the log4j2.xml file in the config directory. Your log4j2.xml file may have a slightly different layout than shown here, but it will contain the four fields highlighted in the below figure. The fields you may wish to edit are:

   ```
   fileName="/opt/keyfactor-java-agent/logs/CMS-Java.txt"
   ```

   The path and file name of the active Java Agent log file.

   > **Important:** If you choose to change the path for storage of the log files, you will need to create the new directory (e.g. /opt/javalogs) and grant the service account under which the Keyfactor Java Agent service is running full control permissions on this directory.

   ```
   size="3 MB"
   ```

   The maximum file size of each log file. After a log file reaches the maximum size, it is rotated to an archive file name and a new log file is generated.

   ```
   max="7"
   ```

   The number of archive files to retain before deletion.

   ```
   level="info"
   ```

   The level of log detail that should be generated. The default INFO level logs error and some informational data but at a minimal level to avoid generating large log files. For troubleshooting, it may be desirable to set the logging level to DEBUG or TRACE. Available log levels (in order of increasing verbosity) are:

- OFF – No logging
- FATAL – Log severe errors that cause early termination
- ERROR – Log severe errors and other runtime errors or unexpected conditions that may not cause early termination
- WARN – Log errors and use of deprecated APIs, poor use of APIs, almost errors, and other runtime situations that are undesirable or unexpected but not necessarily wrong
- INFO – Log all of the above plus runtime events (startup/shutdown)
- DEBUG – Log all of the above plus detailed information on the flow through the system
- TRACE – Maximum log information—this option can generate VERY large log files

```
<RollingFile name="logfile" fileName="/opt/cms-java-agent/logs/CMS-Java.txt" filePattern="CMS-Java-%i.txt">
  <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
  <MarkerFilter marker="EVENT" onMatch="DENY" onMismatch="NEUTRAL" />
  <SizeBasedTriggeringPolicy size="3 MB"/>
  <DefaultRolloverStrategy max="7"/>
</RollingFile>
Section of file removed in graphic for simplicity.
<Root level="info">
  <AppenderRef ref="console" />
  <AppenderRef ref="logfile" />
```

Figure 591: Configure Logging for Keyfactor Java Agent on Linux

5.3.4.2 Start the Keyfactor Java Agent Service

The Keyfactor Java Agent service runs on the Java Agent machine and controls discovery, inventory and certificate store update tasks. During the Java Agent configuration process you set the service account under which the Keyfactor Java Agent service will run. The service should start automatically at the conclusion of the installation scripts.

To check to see if the Keyfactor Java Agent service is running and start it if necessary on Windows:

1. On a Windows Java Agent server, open the Services MMC.
2. In the Services MMC confirm that the Keyfactor Java Agent service is set to a Startup Type of Automatic (if desired). If the service is not running, click the green arrow to start it.

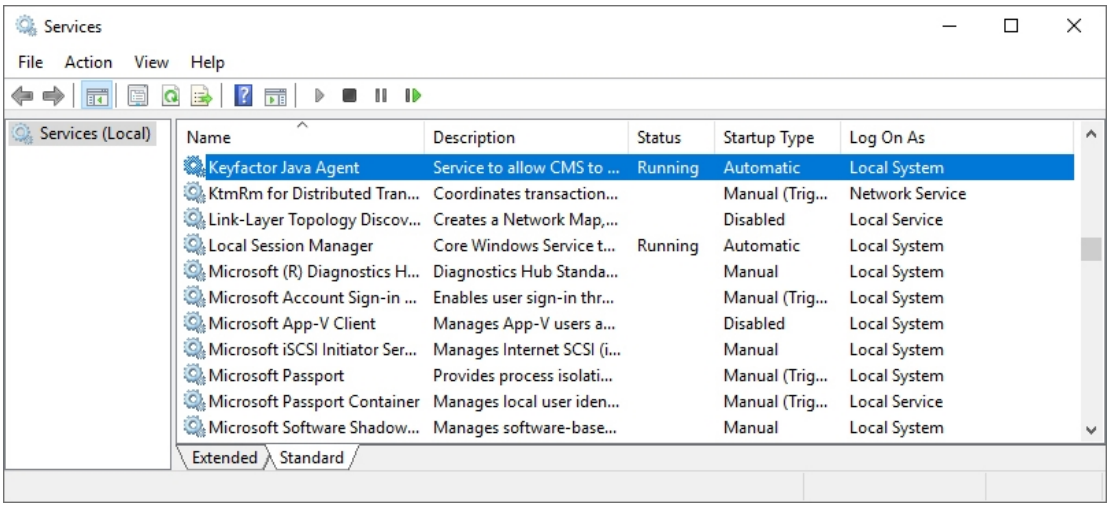


Figure 592: Keyfactor Java Agent Service on Windows

Service startup and shutdown procedures vary by Linux implementation and version depending on the startup system. The service on Linux is added as keyfactor-java-agent, so when referencing it in startup commands, it should be referenced by this name, including case. For example:

```
service keyfactor-java-agent start [stop] [restart]
systemctl start [stop] [restart] [status] keyfactor-java-agent.service
```

Once you have finished the Java keystore and PEM certificate store inventory configuration using the Keyfactor Command Management Portal and have imported certificates from the stores, you can use the Certificate Search feature in the Keyfactor Command Management Portal to review the certificate store certificates. See Certificate Search and Collections on page 19 in the *Keyfactor Command Reference Guide* for information on using the Certificate Search feature.

5.3.4.3 Uninstall the Java Agent

To uninstall the Java Agent on Linux.

1. On the Linux machine on which the Java Agent is installed, run the command to stop the service.

   ```
   sudo systemctl stop keyfactor-java-agent.service
   ```

2. Run the command to remove the service

   ```
   sudo systemctl disable keyfactor-java-agent.service
   ```

3. After steps 1 & 2 are executed, it is safe to manually remove the Java Agent folder (default location is /opt/keyfactor-java-agent/).

5.4 Bash Orchestrator

SSH supports a wide variety of authentication mechanisms. Often, enterprises fall back to simple username and password at least some of the time due to the complexities of key management for key-based authentication. Without key management, SSH keys tend to multiply, and you can quickly lose track of who has access to what where. The Keyfactor Bash Orchestrator is designed to allow organizations to inventory and manage secure shell (SSH) keys across the enterprise.

> **Important:** SSH Key Management licensing is required to use any of the functionality outlined in the Keyfactor Bash Orchestrator documentation. Contact support@keyfactor.com for assistance with obtaining the proper licenses.

The orchestrator runs on Linux servers and can be operated in two possible modes:

- The orchestrator is used in *inventory only* mode to perform discovery of SSH public keys and associated Linux user accounts across multiple configured targets. When used in *inventory and publish policy* mode, the orchestrator:
  - Scans the authorized_keys files of all current users on each configured server.

    > **Note:** OpenSSH maintains a file for each user that contains the public keys authorized to connect via SSH. By default, this file is named authorized_keys. In this document, we refer to this file as *authorized_keys*, however in your environment, this file may have a different name. The file name used in a given environment is defined in the AuthorizedKeysFile setting in the OpenSSH sshd_config file.

  - Aggregates all public key data per Linux user logon.
  - Reports aggregate key and logon data back to Keyfactor Command.

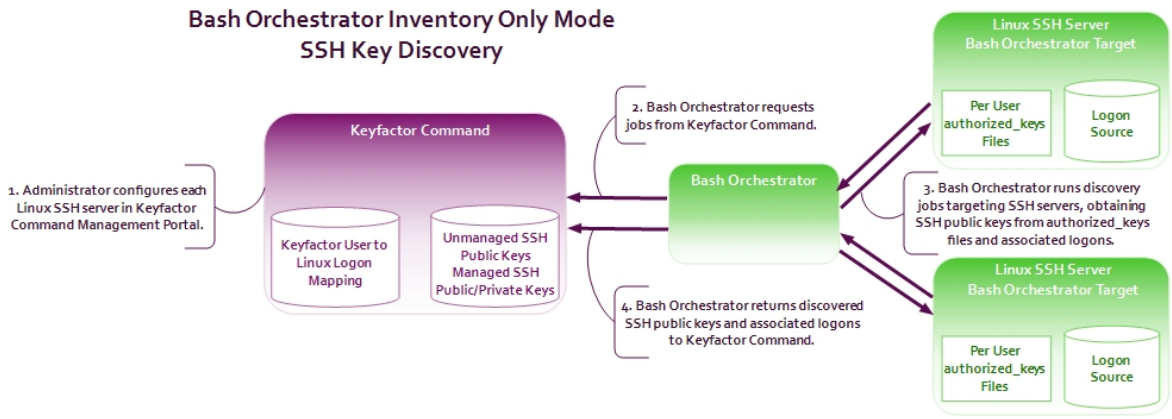


Figure 593: SSH Key Discovery Flow

- When operated in *inventory and publish policy* mode, the orchestrator can be used to add SSH public keys and Linux user accounts on targets and remove rogue keys that appear without authorization. Figure 594: SSH User Key Management Flow shows the flow from a user

requesting a new key pair to the public key being placed on a target server to allow the user to connect to the server via SSH. The flow is similar for requesting a key pair for a service, though the request is made by an administrator through a different interface in the Keyfactor Command Management Portal. When used in *inventory and publish policy* mode, policies are published to the orchestrator from the Keyfactor Command server following this flow:

- The Keyfactor Command server determines what content needs to go into the authorized_keys files for each logon on each target server. Content includes keys and associated comments aggregated from all servers where that key was found. For example, if a given public key exists on three different servers for the same user but in the original authorized_keys files the key is associated with a different comment on each server, when Keyfactor Command publishes the key down to the servers, it will be published with an aggregated comment string (all three comments together in each authorized_keys file).
- Aggregate logon and key information pushed down to each orchestrator target.
- Orchestrator determines where to place key information, builds the file, and overwrites the existing file with the new one. The process is done in this way to enforce policy and prevent rogue keys from being placed in authorized_keys files.
- Orchestrator informs Keyfactor Command of the success or failure of each machine logon combination.

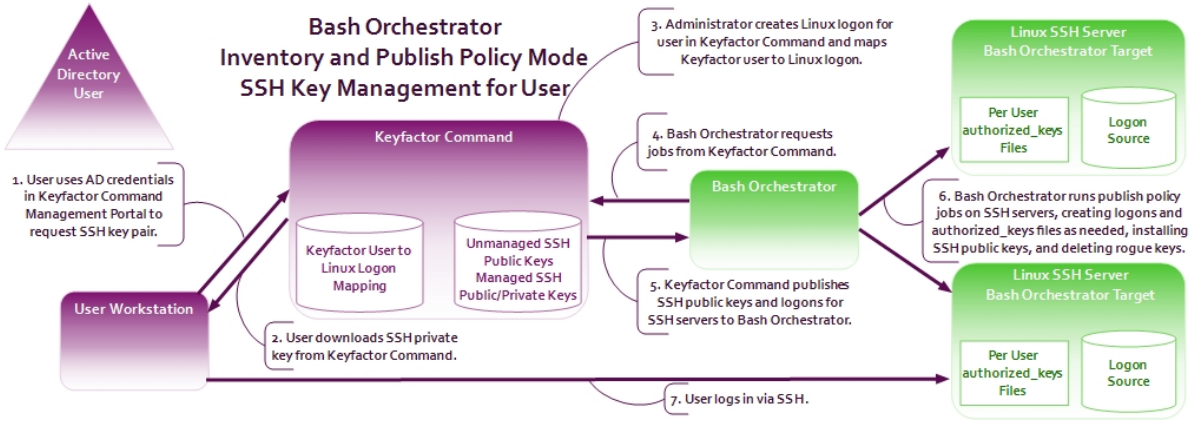


Figure 594: SSH User Key Management Flow

5.4.1 Preparing for the Keyfactor Bash Orchestrator

This section describes the steps that need to be taken prior to a Keyfactor Bash Orchestrator installation to install the prerequisites, create the required supporting components, and gather the necessary information to complete the orchestrator installation and configuration process.

5.4.1.1 System Requirements

The Keyfactor Bash Orchestrator is supported on the following operating systems:

- Oracle Linux 7 or higher
- Red Hat Enterprise 7 or higher
- Ubuntu 16 or higher

The minimum specs are:

- 2GB RAM
- 1 2GHz Processor
- 20GB disk space

Note: As more servers are added to be orchestrated by the orchestrator, increasing those specs will improve performance.

The following applications are required:

systemd

The orchestrator requires a Linux server that uses the systemd service manager. You can use the following command to test whether a system is running systemd:

```
ps -p 1
```

bash

The orchestrator can only be installed on a Linux server that is running bash version 4.3 or higher. You can use the following command to check the bash version of a server:

```
bash --version
```

For systems running an older version of bash, you may be able to successfully operate these as control targets (see Install Remote Control Targets on page 3000).

Note: The default version of bash on Red Hat Enterprise 7 is 4.2. If you're using this platform and your bash version has not already been updated, this will need to be done.

curl

The orchestrator can only be installed on a Linux server that has curl installed. You can use the following command to check the curl version of a server:

```
curl --version
```

This is a requirement for orchestrators only; curl does not need to be installed on control targets (see Install Remote Control Targets on page 3000).

5.4.1.2 Create a Service Account for the Keyfactor Bash Orchestrator

The Keyfactor Bash Orchestrator uses a service account in the Active Directory domain where the Keyfactor Command server resides to allow it to communicate with Keyfactor Command. This can be the same service account used for other Keyfactor Command server services. This service account appears in the Management Portal as the *Identity* on the Orchestrator Management grid for the Keyfactor Bash Orchestrator.

The service account needs to be created prior to installation of the Keyfactor Bash Orchestrator software, and the person installing the Keyfactor Bash Orchestrator software needs to know the domain, username and password of the service account.

> **Important:** Keyfactor highly recommends that you use strong passwords for any accounts or certificates related to Keyfactor Command and associated products, especially when these have elevated or administrative access. A strong password has at least 12 characters (more is better) and multiple character classes (lowercase letters, uppercase letters, numeral, and symbols). Ideally, each password would be randomly generated. Avoid password re-use.

During installation of the orchestrator, a local Linux user account is created automatically as an identity under which the orchestrator service will operate. This allows the orchestrator to run as a non-root user. On servers on which you install the orchestrator directly, the following Linux user account is created:

keyfactor-bash

On servers configured as remote control targets, the following Linux user account is created:

keyfactor-bash-orchestrator-svc

These users are granted access to read authorized_keys files for inventory purposes and to update authorized_keys files when the orchestrator is operating in *inventory and publish policy* mode using sudo. On install, modifications are made to the sudo configuration with the addition of a file in the /etc/sudoer.d directory granting the orchestrator user select sudo rights. The commands the service account user may be granted the right to use via sudo include:

```
adduser, awk, cat, chmod, chown, flock, gpasswd, ls, mkdir, restorecon, rm, sed, tee,
test, touch, usermod
```

5.4.1.3 Create a Group for Auto-Registration (Optional)

Keyfactor Command can use an Active Directory group to support auto-registration of Keyfactor Bash Orchestrator. Auto-registration is an optional feature that allows you to define the conditions under which a Keyfactor Bash Orchestrator can automatically be approved for operation with the Keyfactor Command server without administrator input, if desired. This is useful in environments hosting a large number of orchestrators or if you wish to automatically add orchestrators to server

groups and add them as servers in the Management Portal as you install them. The auto-registration role used by the Keyfactor Bash Orchestrator is called *Secure Shell Management*.

Add the service account or service accounts under which the orchestrators will communicate with Keyfactor Command to this group.

Note: If all your orchestrators will be connecting to Keyfactor Command as the same service account, you can directly add that user in the auto-registration configuration and skip using a group, if desired.
Although you can choose to enable auto-registration without user validation, allowing any orchestrator to register regardless of the user account under which the orchestrator is running, user validation with either an Active Directory group or a specific Active Directory user is the more secure option.

5.4.1.4 Certificate Root Trust for the Keyfactor Bash Orchestrator

Keyfactor recommends using HTTPS to secure the channel between each Keyfactor Bash Orchestrator and the Keyfactor Command server(s). This requires an SSL certificate configured in IIS on the Keyfactor Command server(s). This certificate can either be a publicly-rooted certificate (e.g. from Symantec, Entrust, etc.), or one issued from a private certificate authority (CA). If your Keyfactor Command server is using a publicly rooted certificate, the orchestrator machine may already trust the certificate root for this certificate. However, if you have opted to use an internally-generated certificate, your orchestrator server may not trust this certificate. In order to use HTTPS for communications between the orchestrator and the Keyfactor Command server with a certificate generated from a private CA, you will need to import the certificate chain for the certificate into the orchestrator's root certificate store. This can be done automatically as part of the installation process. You will need to have the root certificate available as a PEM-encoded format file when you run the installation script.

5.4.2 Install the Keyfactor Bash Orchestrator

To begin the Keyfactor Bash Orchestrator installation, place the installation files in a temporary working directory on the Linux server and:

1. On the Linux machine on which you wish to install the main orchestrator, in a command shell change to the temporary directory where you placed the installation files.
2. Use the chmod command to make the following script files executable. The files ship in a non-executable state to avoid accidental execution.
    - [yourpath]/heartbeat.sh
    - [yourpath]/static-analysis.sh
    - [yourpath]/syncjob.sh
    - [yourpath]/Service/keyfactor-bash-orchestrator.sh
    - [yourpath]/Service/systemd/configure-systemd.sh

- [yourpath]/Service/systemd/stop.sh
- [yourpath]/Installation/install.sh
- [yourpath]/Installation/remoteinstall.sh
- [yourpath]/Installation/uninstall.sh

For example, this command will add the executable flag to every file with a .sh extension in the /tmp/BashOrchestrator directory and all its sub-directories:

```
sudo find /tmp/BashOrchestrator -type f -iname "*.sh" -exec chmod +x {} \;
```

3. In the command shell, run the Installation/install.sh script as root using the following syntax to begin the installation:

-n, --username <*service account name*>

This is the service account that the orchestrator uses to communicate with Keyfactor Command that you created as per Create a Service Account for the Keyfactor Bash Orchestrator on page 2994. It should be entered in the format username@domain (e.g. svc_sshorch@keyexample.com). This parameter is required.

-u, --url <*Keyfactor Command agents URL*>

This is the URL to the Agent Services endpoint on the Keyfactor Command server running the Keyfactor Command Agent Services role, which is installed as part of the Keyfactor Command Services role. If you installed all the Keyfactor Command server roles together, this is the URL for your Keyfactor Command server with /KeyfactorAgents after the server's IP or FQDN (e.g. https://keyfactor.keyexample.com/KeyfactorAgents). If you choose to use SSL to connect to the Keyfactor Command server, you'll need to enter a URL that contains a hostname that is found in the SSL certificate. This parameter is required.

Tip: If your Keyfactor Command server was configured with an alternate virtual directory for the Keyfactor Command Agents Services endpoint, you will need to enter that in the URL rather than /KeyfactorAgents.

-p, --password <*your-secure-password*>

This is the password for the orchestrator service account. If you leave this parameter out, you will be prompted to enter this password.

-s, --ssl

Specifying this parameter causes the orchestrator to use SSL for communications with Keyfactor Command. Leave out this parameter if you prefer not to use SSL. This parameter does not take any arguments.

-t, --trusted-root </path/root-filename>

If your Keyfactor Command server is using a publicly rooted certificate, you do not need to use this option. If the certificate on the Keyfactor Command server was internally generated, you will need to use this option to specify the full path and file name of the file containing the root certificate for the certificate authority that issued the certificate (e.g. -t /tmp/myroot.crt). See Certificate Root Trust for the Keyfactor Bash Orchestrator on page 2995.

-i, --server-group-id <*GUID of existing SSH server group*>

If desired, you may specify this parameter to automatically add the server to an existing server group in Keyfactor Command. The server group must be specified by group ID (e.g. -i 74a9afcc-087d-423a-a331-06686427fdd9). You can find a server group's ID by editing the server group record in the Keyfactor Command Management Portal. This function is only supported if you have enabled auto-registration for SSH (see Create a Service Account for the Keyfactor Bash Orchestrator on page 2994).

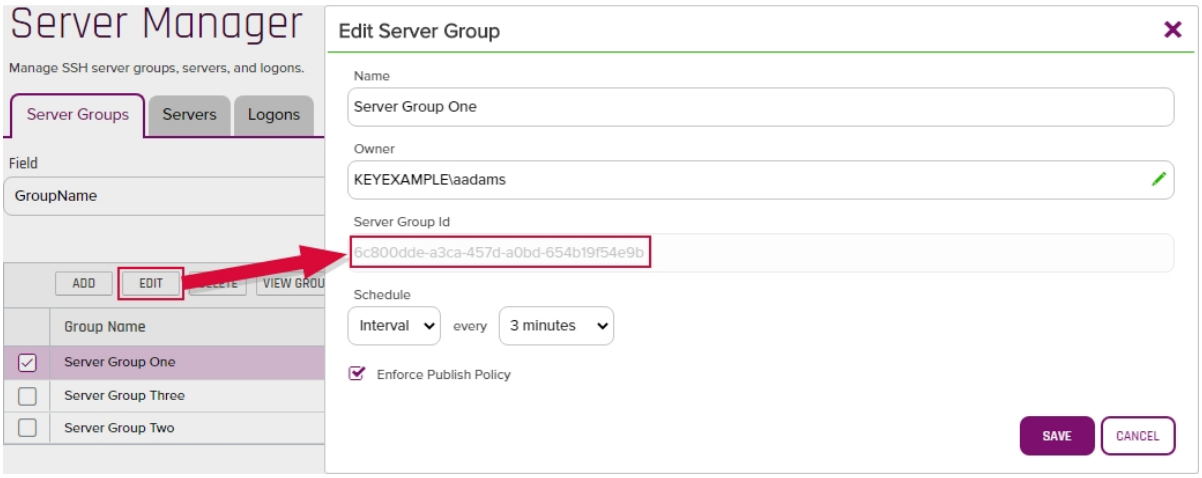


Figure 595: Find the Server Group ID

-c, --client-machine-name <*client name*>

Specifying this parameter allows you to override the client name the orchestrator would by default use to register itself in Keyfactor Command. By default, the orchestrator uses the results from a hostname lookup for the server's name. See the example output below where the name passed into Keyfactor Command (appsrvr80-SSH.ubuntu.keyexample.com) differs from the name used in the SSH key comment for the local Linux user (appsrvr80.keyexample.com).

-d, --use-sssd

This is required. You must explicitly specify whether or not you want to enable the orchestrator to use SSSD for user lookups.

- To enable SSSD set either:

```
-d true
```

```
--use-sssd true
```

- To disable SSSD set either:

```
-d false
```

```
--use-sssd false
```

When enabled, the orchestrator will check both the local user store and the SSSD user store (e.g. Active Directory) on requests to create logons and distribute key information, allowing keys to be managed both for local users and for domain users. When enabled, user logon must be created in Keyfactor Command with the username as it appears in SSSD (see SSH-SSSD Case Sensitivity Flag on page 791 and Adding Logons on page 585).

If you're using SSSD, you must be using SSSD on any remote servers the orchestrator will manage. Additionally, the *LogonHomeDirectories* setting is expected to be consistent on all remote servers.

Domain users can be managed with or without preexisting home directories.

-l, --logon-home-directories </*homedirectoryroot*>

Specifying this parameter allows you to set the base path for home directories of SSH users. This is referenced both when new logons are created, as requested through Keyfactor Command, and when doing discovery for existing logons and keys. If you don't specify a value, the default of */home* is used.

The value set for the Keyfactor Bash Orchestrator *login-home-directories* needs to match the value set for the path in the *override_homedir* or *fallback_homedir* SSSD configuration. For example, if *fallback_homedir* = /home/my/dir/path/%u@%d, *login-home-directories* needs to be set to /home/my/dir/path. All SSSD logons to be discovered by or created with the Keyfactor Bash Orchestrator must have a home directory in this directory, not a subdirectory of this directory. For example, given the previously referenced directory, the path /home/my/dir/path/myusername@keyexample.com would be valid but /home/my/dir/path/anotherdirlevel/myusername@keyexample.com would not be valid. Home directories are created automatically when logons are created.

> ⚠ **Important:** Any remotely controlled targets (see Install Remote Control Targets on page 3000) of a server using SSSD logons with the Keyfactor Bash Orchestrator must also be configured for SSSD logons and must have the same configuration value for *fallback_homedir* or *override_homedir*.

The output from the command should look similar to the following, given the example command shown.

```
sudo ./install.sh -u https://keyfactor.keyexample.com/KeyfactorAgents -n svc_
sshorch@keyexample.com -s -t /tmp/MyRoot.crt -i 74a9afcc-087d-423a-a331-06686427fdd9 -c
appsrvr80-SSH.ubuntu.keyexample.com -d false

Service Account Password:
Creating orchestrator installation directory...
Creating file structure...
Generating public/private rsa key pair.
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:APQcjxNSzPFF0Dg+cFlteJjHb2CoIAK7/ysAkUkKk7s keyfactor-bash@appsrvr80.keyexample.com
The key's randomart image is:
+---[RSA 2048]----+
|=* .++=..B+B     |
|Bo. .==*=.* O    |
|oo . .*=oo = o   |
|o.     o+     o  |
|o.      S.   .   |
|E.               |
| ..              |
|  ..             |
|   .o.           |
+----[SHA256]-----+
Creating orchestrator log file...
Creating Session Cache File...
Adding uninstall script to installation directory...
Installing Keyfactor Bash Orchestrator...
Creating credential file...
Creating job schedule table...
Adding root certificate to local ca store...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

done.
done.
Creating Keyfactor SSH Daemon...
Creating service unit file...
Setting file ownership...
Ensuring service account 'keyfactor-bash' has necessary permissions...
```

```
Creating remote setup script...
Starting Keyfactor Bash Orchestrator...
```

4. Review the output from the installation to confirm that no errors have occurred.

The script creates a directory, /opt/keyfactor-bash-orchestrator, and places the orchestrator files in this directory. Log files are found in /opt/keyfactor-bash-orchestrator/logs, though this is configurable (see Configure Logging for the Keyfactor Bash Orchestrator on page 3002).

The orchestrator service, named keyfactor-bash-orchestrator.service, should be automatically started at the conclusion of the install and configured to restart on reboot.

Tip: Once the installation of the orchestrator and any targets for it to control is complete, you need to use the Keyfactor Command Management Portal to approve the orchestrator (if you don't have auto-registration for Keyfactor Bash Orchestrators enabled) and configure SSH server groups and servers as per Server Manager on page 560 in the *Keyfactor Command Reference Guide*. SSH server records are automatically created for the main bash orchestrator if you enable auto-registration for bash orchestrators and use the -i switch when registering the bash orchestrator. They are not automatically created for remote targets.

5.4.3 Install Remote Control Targets

After you complete the installation of at least one Keyfactor Bash Orchestrator, you can configure other Linux servers in the environment as control targets for this orchestrator. This is done by running a script on the target servers that installs the SSH public key matching the orchestrator's private key on the target server, along with making a few configuration changes. This allows the orchestrator service on the orchestrator (the local Linux user keyfactor-bash) to communicate with the targets using secured SSH.

Important: Any remotely controlled targets of a server using SSSD logons with the Keyfactor Bash Orchestrator must also be configured for SSSD logons and must have the same configuration value for *fallback_homedir* or *override_homedir*.

To configure orchestrator targets:

1. On the orchestrator machine, locate the remoteinstall.sh script in the /opt/keyfactor-bash-orchestrator directory. Do not use the remoteinstall-template.sh script found in the source material under Installation. This script has not been modified to contain the specific public key of your orchestrator.

   Tip: A copy of the configured remoteinstall.sh script may also be found in the directory from which you executed the installation of the Keyfactor Bash Orchestrator.

2. Copy the customized remoteinstall.sh script to the orchestrator target that you wish to configure and place it in a temporary working directory.

3. On the Linux machine you wish to control with the orchestrator, in a command shell change to the temporary directory where you placed the remoteinstall.sh script.

4. Use the chmod command to make the script file executable. The file ships in a non-executable state to avoid accidental execution. For example:

   ```
   sudo ./chmod +x remoteinstall.sh
   ```

5. In the command shell, run the remoteinstall.sh script as root with no parameters. There is no output from the command when it completes successfully.

   ```
   sudo ./remoteinstall.sh
   ```

The script creates a directory, /opt/keyfactor-bach-orchestrator-client, and places the public key of the orchestrator Linux service account user in an authorized_keys file within it. It also creates a local service account user (see Create a Service Account for the Keyfactor Bash Orchestrator on page 2994) and grants this user ownership on this file to allow the orchestrator server service account to perform tasks on the target.

Log messages are written to the standard Linux syslog. The location of these will vary depending on the system OS.

Tip: Once the installation of the orchestrator and any targets for it to control is complete, you need to use the Keyfactor Command Management Portal to approve the orchestrator (if you don't have auto-registration for Keyfactor Bash Orchestrators enabled) and configure SSH server groups and servers as per Server Manager on page 560 in the *Keyfactor Command Reference Guide*. SSH server records are not automatically created for remote targets, even if you enable auto-registration for bash orchestrators and use the -i switch when registering the bash orchestrator that will control your targets.

5.4.4 Optional Configuration

Once the installation is complete, the Keyfactor Bash Orchestrator should be running and ready to communicate with the Keyfactor Command server. The initial installation allows the orchestrator to scan itself to do discovery of SSH keys and then management of SSH keys if the server is configured for management in Keyfactor Command. At this point, you may wish to configure one or more orchestrator target servers for the orchestrator to additionally control (see Install Remote Control Targets on the previous page).

Important: Orchestrator tasks will not run until you complete the orchestrator configuration by making the appropriate configuration changes in the Keyfactor Command Management Portal. See Orchestrators on page 481 in the *Keyfactor Command Reference Guide* for instructions on approving the orchestrator in the Keyfactor Command Management Portal on the *Orchestrators->Management* pages and on configuring SSH server groups and servers

on the *SSH->Server Managers* page (see Server Manager on page 560 in the *Keyfactor Command Reference Guide*).

5.4.4.1 Configure Logging for the Keyfactor Bash Orchestrator

By default, the Keyfactor Bash Orchestrator places its log files in the /opt/keyfactor-bash-orchestrator/logs directory, generates logs at non-debug level, rotates the logs when they reach 50 MB, and retains 10 archive logs before deletion.

If you wish to change these defaults after the installation is complete:

1. On the orchestrator machine where you wish to adjust logging, open a command shell and change to the directory in which the orchestrator is installed. By default this is /opt/keyfactor-bash-orchestrator.
2. In the command shell in the directory in which the orchestrator is installed, change to the Configuration directory.
3. Using a text editor, open the orchestrator_config file in the Configuration directory. Your orchestrator_config file may have a slightly different layout than shown here, but it will contain the three fields highlighted in the below figure. The fields you may wish to edit are:
    - logFile=/opt/keyfactor-bash-orchestrator/logs/bash-orchestrator-log.txt

      The path and file name of the active orchestrator log file.

      Important: If you choose to change the path for storage of the log files, you will need to create the new directory (e.g. /opt/sshorchlogs) and grant the Linux service account under which the orchestrator service is running (see Create a Service Account for the Keyfactor Bash Orchestrator on page 2994) full control permissions on this directory.

    - logFileSize=50000000
      The maximum file size of each log file. After a log file reaches the maximum size, it is rotated to an archive file name and a new log file is generated. The default is 50000000 (50 MB).
    - logFilesToKeep=10
      The number of archive files to retain before deletion.
    - debugLogEnabled=false
      The level of log detail that should be generated. The default of *false* logs error and some informational data but at a minimal level to avoid generating large log files. For troubleshooting, it may be desirable to set the debug level to *true*.

```
logFile=/opt/keyfactor-bash-orchestrator/logs/bash-orchestrator-log.txt
logFileSize=50000000
logFilesToKeep=10
keyfactorAgentServer=https://keyfactor.keyexample.com/KeyfactorAgents
serverGroupId=71804a75-78de-42fb-bb7e-96b123742f0d
debugLogEnabled=false
clientMachineName=appsrvr79.keyexample.com
```

Figure 596: Configure Logging for the Keyfactor Bash Orchestrator

Tip: Log messages for remote control targets are written to the standard Linux syslog. The location of these will vary depending on the system OS. Log messages for the orchestrator's communication with the remote control targets are included in the primary orchestrator log (described above). It can be helpful to look in both places when troubleshooting an issue with a remote control target.

5.4.4.2 Start the Keyfactor Bash Orchestrator Service

The keyfactor-bash-orchestrator service runs on the Keyfactor Bash Orchestrator machine and controls SSH public key discovery and management tasks for the orchestrator machine itself and target servers it controls. The service should start automatically at the conclusion of the installation.

The service on Linux is added as keyfactor-bash-orchestrator, so when referencing it in startup commands, it should be referenced by this name, including case. For example:

```
systemctl start [stop] [restart] [status] keyfactor-bash-orchestrator.service
```

Once you have finished the SSH server group and server configuration using the Keyfactor Command Management Portal and have completed a scan of the configured servers, you can view discovered keys and logons in the Keyfactor Command Management Portal and then begin using the management features. See SSH on page 525 in the *Keyfactor Command Reference Guide* for information on using the SSH features.

5.5 Troubleshooting

The following error conditions and general troubleshooting tips may be helpful in resolving issues with the Keyfactor orchestrators. Generally speaking, issues are often related to trusts of root and intermediate certificates, firewall challenges, or insufficient permissions for the service account running the orchestrator service.

Validate Management Portal Configuration

Things to check in the Management Portal include:

- Is the last seen time for the orchestrator on the Orchestrator Management page in the Management Portal within the last few minutes (see Orchestrator Management on page 496 in the *Keyfactor Command Reference Guide*)? Most orchestrators send a heartbeat to Keyfactor Command every 5 minutes, so this date should at most be 5 minutes out of date if the

orchestrator is operating correctly.

> **Tip:** Orchestrator control targets for the Keyfactor Bash Orchestrator do not appear on the Orchestrator Management page, so for a remote server that's not operating as expected, this would be the orchestrator that is controlling the target.

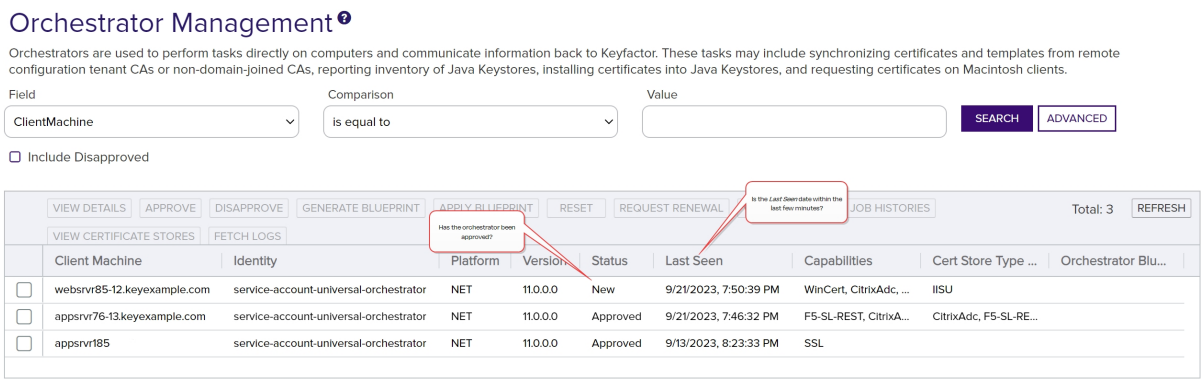


Figure 597: Orchestrator Management for a Keyfactor Bash Orchestrator

- Has the orchestrator been approved on the Orchestrator Management page in the Management Portal (see Orchestrator Management on page 496 in the *Keyfactor Command Reference Guide*)?
- Is there a sync schedule set to run frequently for the orchestrator (SSH), remote control target (SSH), or certificate store? Sync schedules for certificates stores are automatically disabled if inventory jobs are failing.
- For the Keyfactor Bash Orchestrator:
  - Has the server record for the orchestrator or remote control target been created under SSH Server Manager on the Servers tab in the Management Portal (see SSH Servers on page 577 in the *Keyfactor Command Reference Guide*)?

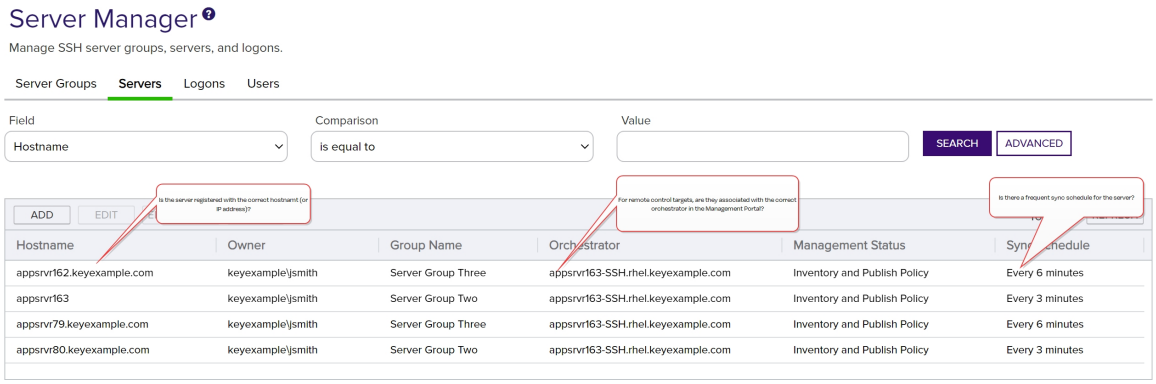


Figure 598: Orchestrator Management for a Keyfactor Bash Orchestrator

- Does the server record for the orchestrator or remote control target in the Management Portal have the correct hostname or IP address? If the name or IP address is incorrect, sync jobs will fail.
- Is the server record for the remote control target in the Management Portal associated with the correct orchestrator? If the control target is associated with the wrong orchestrator, you may be looking at the wrong log files (see Debug Logging and Error Messages below) for troubleshooting information.

Debug Logging and Error Messages

It is often helpful to enable debug logging on the orchestrator. For information on configuring this, see the specific orchestrator chapters.

Once the logging is set at debug or trace level, it can be helpful to watch the logs live while activity is going on. On Linux, you can do this with *tail* (or a similar tool) to watch the log in real time. For example:

```
tail -f /opt/keyfactor-bash-orchestrator/logs/keyfactorbash-orchestrator-log.txt
```

```
tail -f /opt/keyfactor/orchestrator/logs/Log.txt
```

On Windows, there are also tools with tail-like functionality. Notepad++, for example, has this functionality built in.

Some messages in the KeyfactorUniversal Orchestrator log include a correlation ID that helps to identify log messages that originated from the same request. The correlation ID is a randomly generated GUID that often appears just after the date in the log entry (B0C4946E-DB3B-4404-8080-79AFF260DE4E in the following example) and is the same for all log messages for the given request until the request completes.


```
2023-09-15 18:19:00.3780 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398 Keyfactor.Orches-
trators.JobExecutors.OrchestratorJobExecutor [Debug] - Running job extension for  job with Id
'b0c4946e-db3b-4404-8080-79aff260de4e'
2023-09-15 18:19:00.3780 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398 Keyfactor.Ex-
tensions.Orchestrator.WindowsCertStore.WinCert.Inventory [Trace] - Entered 'ProcessJob' method.
2023-09-15 18:19:00.3780 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398 Keyfactor.Ex-
tensions.Orchestrator.WindowsCertStore.WinCert.Inventory [Trace] - {"JobCan-
celled":-
false,"ServerError":null,"JobHistoryID":230398,"RequestStatus":1,"ServerUserName":"keyexample\\svc_
kyforch","ServerPassword":"**********","JobConfigurationProperties":{"spnwithport":false,"WinRm
Protocol":"https","WinRm Port":"5986","ServerUsername":"keyexample\\svc_kyforch","Server-
UseSsl":true,"sniflag":0},"UseSSL":true,"JobTypeID":"00000000-0000-0000-0000-
000000000000","JobID":"b0c4946e-db3b-4404-8080-79aff260de4e","Cap-
ability":"CertStores.WinCert.Inventory","LastInventory":[],"CertificateStoreDetails":
{"ClientMachine":"web-
srvr93.keyexample.com","StorePath":"My","StorePassword":"**********","Type":117}}
```

```
2023-09-15 18:19:00.3780 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398 Keyfactor.Ex-
tensions.Orchestrator.WindowsCertStore.WinCert.Inventory [Trace] - Establishing runspace on client
machine: websrvr93.keyexample.com
2023-09-15 18:19:00.3780 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398 Keyfactor.Ex-
tensions.Orchestrator.WindowsCertStore.PsHelper [Trace] - Entered 'GetClientPsRunspace' method.
2023-09-15 18:19:00.3780 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398 Keyfactor.Ex-
tensions.Orchestrator.WindowsCertStore.PsHelper [Trace] - Creating remote session at: https://web-
srvr93.keyexample.com:5986/wsman
2023-09-15 18:19:00.3780 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398
Keyfactor.Extensions.Orchestrator.WindowsCertStore.PsHelper [Trace] - Credentials Specified
   [Messages removed for clarity]
2023-09-15 18:19:00.7389 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398 Keyfactor.Ex-
tensions.Orchestrator.WindowsCertStore.WinCert.Inventory [Trace] - Connecting to remote server websr-
vr93.keyexample.com failed with the following error message : acquiring creds with username only
failed No credentials were supplied, or the credentials were unavailable or inaccessible SPNEGO
cannot find mechanisms to negotiate For more information, see the about_Remote_Troubleshooting Help
topic.
   at System.Management.Automation.Runspaces.AsyncResult.EndInvoke()
   at System.Management.Automation.Runspaces.Internal.RunspacePoolInternal.EndOpen(IAsyncResult asyn-
cResult)
   at System.Management.Automation.Runspaces.Internal.RemoteRunspacePoolInternal.Open()
   at System.Management.Automation.Runspaces.RunspacePool.Open()
   at System.Management.Automation.RemoteRunspace.Open()
   at Keyfactor.Extensions.Orchestrator.WindowsCertStore.WinCert.Inventory.PerformInventory(Invent-
oryJobConfiguration config, SubmitInventoryUpdate submitInventory)

2023-09-15 18:19:00.7389 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398 Keyfactor.Ex-
tensions.Orchestrator.WindowsCertStore.WinCert.Inventory [Warn] - Inventory job failed for Site 'My'
on server 'websrvr93.keyexample.com' with error: 'Connecting to remote server websr-
vr93.keyexample.com failed with the following error message : acquiring creds with username only
failed No credentials were supplied, or the credentials were unavailable or inaccessible SPNEGO
cannot find mechanisms to negotiate For more information, see the about_Remote_Troubleshooting Help
topic.
   at System.Management.Automation.Runspaces.AsyncResult.EndInvoke()
   at System.Management.Automation.Runspaces.Internal.RunspacePoolInternal.EndOpen(IAsyncResult asyn-
cResult)
   at System.Management.Automation.Runspaces.Internal.RemoteRunspacePoolInternal.Open()
   at System.Management.Automation.Runspaces.RunspacePool.Open()
   at System.Management.Automation.RemoteRunspace.Open()
   at Keyfactor.Extensions.Orchestrator.WindowsCertStore.WinCert.Inventory.PerformInventory(Invent-
oryJobConfiguration config, SubmitInventoryUpdate submitInventory)

2023-09-15 18:19:00.7389 B0C4946E-DB3B-4404-8080-79AFF260DE4E 230398
```

```
Keyfactor.Orchestrators.JobExecutors.OrchestratorJobExecutor [Debug] - Finished running job extension
for  job with Id 'b0c4946e-db3b-4404-8080-79aff260de4e'
```

Some messages to look for include:

- This message (or similar—text varies slight from orchestrator to orchestrator) indicates that the orchestrator has not yet been approved in the Keyfactor Command Management Portal:

  ```
  2021-07-29 09:01:28.5957 Keyfactor.Orchestrators.JobEngine.SessionJobExecutor [Info] - Agent has
  not yet been registered with CMS. Trying again in 30 minutes.
  ```

  After approving the orchestrator in the Management Portal, you can restart the orchestrator service to avoid waiting 30 minutes for the next automated retry.

- Some log message spell out the problem pretty clearly. For example, this message from the Java Agent log:

  ```
  2021-07-29 09:00:02.437 [Scheduler_Worker-1] ERROR com.css_security.cms.JksUtilities - Keystore
  /opt/apps/myapp.jks loaded as type JKS but the provided password is incorrect
  ```

  In this case, the certificate store configuration in the Management Portal is not using the correct password for the store.

- This series of messages in the Java Agent log indicates that the stored credentials file for the Java Agent is no longer useable:

  ```
  2021-07-01 11:24:59.292 [Scheduler_Worker-1] ERROR com.css_secur-
  ity.cms.apache.http.HttpClientFactory - Given final block not properly padded. Such issues can
  arise if a bad key is used during decryption.
  2021-07-01 11:24:59.313 [Scheduler_Worker-1] ERROR com.css_secur-
  ity.cms.apache.http.HttpClientFactory - Could not decrypt credentials file at config/in-
  stall.creds
  2021-07-01 11:24:59.313 [Scheduler_Worker-1] INFO  com.css_
  security.cms.apache.http.HttpClientFactory - Your machine key may have changed. Reencrypt creden-
  tials using local machine key.
  2021-07-01 11:24:59.313 [Scheduler_Worker-1] INFO  com.css_secur-
  ity.cms.apache.http.HttpClientFactory - Generate new credentials by running included cms-creden-
  tial-encryptor utility
  2021-07-01 11:24:59.313 [Scheduler_Worker-1] INFO  com.css_secur-
  ity.cms.apache.http.HttpClientFactory - Try 1. Trying again in 30 seconds
  ```

The credentials file can be recreated to return the Java Agent to functionality (see Appendix A—Generate New Credentials for the Java Agent).

- This series of messages indicates that the Keyfactor Command server is unreachable:

```
2021-07-29 11:59:02.1003 Keyfactor.Orchestrators.JobEngine.SessionClient [Error] - Unable to
heartbeat:
2021-07-29 11:59:02.1003 Keyfactor.Orchestrators.JobEngine.SessionClient [Trace] - Leaving CMSSes-
sionClient.Heartbeat
2021-07-29 11:59:02.1006 Keyfactor.Orchestrators.JobEngine.SessionJobExecutor [Debug] - Heartbeat
success: Unreachable
2021-07-29 11:59:02.1006 Keyfactor.Orchestrators.JobEngine.SessionJobExecutor [Warn] - Heartbeat
endpoint unreachable. Trying again later
```

  This could indicate a network or firewall issue.

- A series of messages similar to this for the Universal Orchestrator can indicate a problem retrieving the CRL for the certificate used to secure the Keyfactor Command server if you've chosen to connect to Keyfactor Command over SSL:

```
2022-09-14 11:15:06.1830   Keyfactor.Orchestrators.JobEngine.SessionJobExecutor [Error] - Error
in SessionManager: Unable to register session.
The SSL connection could not be established, see inner exception.
The remote certificate is invalid because of errors in the certificate chain: Revoc-
ationStatusUnknown, OfflineRevocation
```

  Confirm that the CRLs for the CA that issued the certificate and the remaining CAs in the chain are valid. Confirm that they are available in a location that is accessible to the orchestrator server (e.g. a location other than LDAP if the orchestrator is installed on a server not joined to a domain in the forest where they were issued). If you're using delta CRLs and hosting them on an IIS website using the default CRL suffix as a naming convention (+), be sure to enable double escaping in IIS to allow the orchestrator to retrieve the CRL files containing a plus in the file name.

- Messages that look like errors during SSL scanning are common as attempts are made to connect to TLS endpoints and connections fail or are refused. This is part of the process of testing whether an SSL endpoint is responding and then whether there is a certificate there. Most of these message exist at TRACE level, so monitoring at DEBUG rather than TRACE level will eliminate these messages if they become overwhelming. For example:

```
2022-09-12 10:56:32.3948 EE033BD9-421A-44CA-89BC-10C86949B506 166937 Tls13Probe [Trace] -
Endpoint 192.168.216.87:443 returned status 'ExceptionDownloading' with exception 'System.Ar-
gumentException': The specified nonce is not a valid size for this algorithm. (Parameter 'nonce')
2022-09-12 10:56:39.0567 EE033BD9-421A-44CA-89BC-10C86949B506 166937 Tls13Probe [Trace] -
```

```
Endpoint 192.168.216.158:443 returned status 'ConnectionRefused' with exception 'System.Net.Sock-
ets.SocketException': An existing connection was forcibly closed by the remote host.
2022-09-12 10:57:23.4727 EE033BD9-421A-44CA-89BC-10C86949B506 166937 Tls13Probe [Trace] - Connec-
tion to 192.168.216.87:443 failed
2022-09-12 10:57:24.3345 EE033BD9-421A-44CA-89BC-10C86949B506 166937 a [Trace] - Endpoint
192.168.216.211:443 returned status 'ExceptionDownloading' with exception 'Keyfactor.Orches-
trators.SSL.Pipeline.Exceptions.ConnectionGoneException': Read zero bytes on a blocking read
2022-09-12 10:57:57.9505 EE033BD9-421A-44CA-89BC-10C86949B506 166937 b [Trace] - Endpoint
192.168.216.96:443 returned status 'SslRefused' with exception 'Keyfactor.Orches-
trators.SSL.Pipeline.Exceptions.TlsAlertException': Got TLS alert during TLS handshake: Alert
level 2, Alert description 70
```

Heartbeat

You should see a heartbeat message similar to the following in the log every 5 minutes:

- Keyfactor Universal Orchestrator on Windows:

```
2023-09-12 11:01:16.4598   Keyfactor.Orchestrators.JobEngine.SessionJobExecutor [Debug] -
Existing session found. Heartbeating...
```

- Keyfactor Bash Orchestrator:

```
Tue Aug 11 18:06:02 UTC 2023 [Debug]: Performing orchestrator heartbeat...
```

- Keyfactor Java Agent on Linux:

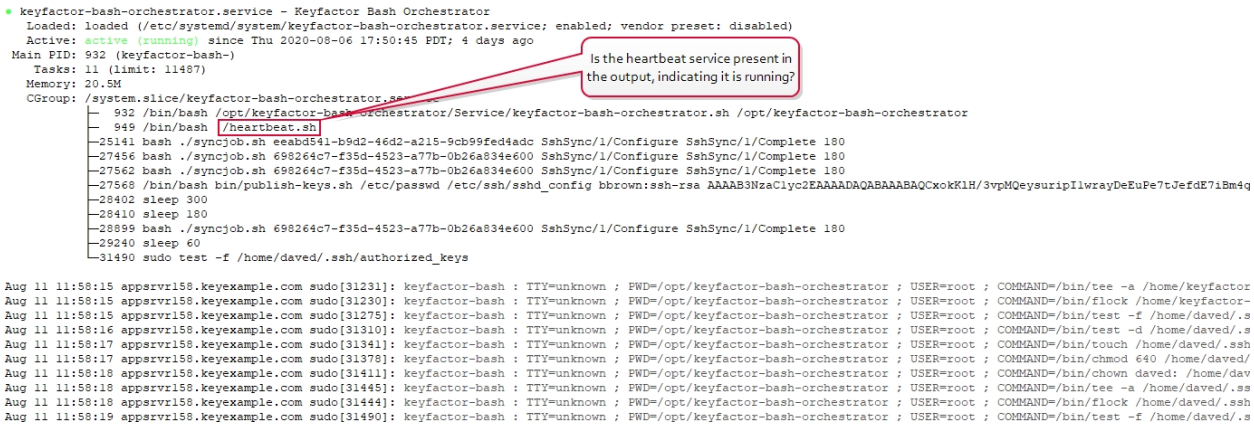
```
2023-07-30 00:52:11.662 [Scheduler_Worker-1] DEBUG com.css_secur-
ity.cms.agents.jobs.SessionManager - Existing session found. Heartbeating...
```

This is the orchestrator checking in with the Keyfactor Command server to see if there are any jobs. If this message is missing, it could indicate that the heartbeat service is not running.

If you're running the Keyfactor Bash Orchestrator, you can see the heartbeat service as a separate entity. Execute this command on the orchestrator in the command shell as root:

```
systemctl status keyfactor-bash-orchestrator.service
```

Output from this command should look something like that shown in Figure 599: Status for the Keyfactor Bash Orchestrator Service. If you don't see heartbeat.sh in the output, the heartbeat service is not running.




```
● keyfactor-bash-orchestrator.service - Keyfactor Bash Orchestrator
   Loaded: loaded (/etc/systemd/system/keyfactor-bash-orchestrator.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2020-08-06 17:50:45 PDT; 4 days ago
 Main PID: 932 (keyfactor-bash-)
    Tasks: 11 (limit: 11487)
   Memory: 20.5M
   CGroup: /system.slice/keyfactor-bash-orchestrator.service
           ├─  932 /bin/bash /opt/keyfactor-bash-orchestrator/Service/keyfactor-bash-orchestrator.sh /opt/keyfactor-bash-orchestrator
           ├─  949 /bin/bash /heartbeat.sh
           ├─25141 bash ./syncjob.sh eeabd541-b9d2-46d2-a215-9cb99fed4adc SshSync/1/Configure SshSync/1/Complete 180
           ├─27456 bash ./syncjob.sh 698264c7-f35d-4523-a77b-0b26a834e600 SshSync/1/Configure SshSync/1/Complete 180
           ├─27562 bash ./syncjob.sh 698264c7-f35d-4523-a77b-0b26a834e600 SshSync/1/Configure SshSync/1/Complete 180
           ├─27568 /bin/bash bin/publish-keys.sh /etc/passwd /etc/ssh/sshd_config bbrown:ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCxokK1H/3vpMQeysuripIlwrayDeEuPe7tJefdE7iBm4q
           ├─28402 sleep 300
           ├─28410 sleep 180
           ├─28899 bash ./syncjob.sh 698264c7-f35d-4523-a77b-0b26a834e600 SshSync/1/Configure SshSync/1/Complete 180
           ├─29240 sleep 60
           └─31490 sudo test -f /home/daved/.ssh/authorized_keys

Aug 11 11:58:15 appsrvr158.keyexample.com sudo[31231]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/tee -a /home/keyfactor
Aug 11 11:58:15 appsrvr158.keyexample.com sudo[31230]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/flock /home/keyfactor-
Aug 11 11:58:15 appsrvr158.keyexample.com sudo[31275]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/test -f /home/daved/.s
Aug 11 11:58:16 appsrvr158.keyexample.com sudo[31310]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/test -d /home/daved/.s
Aug 11 11:58:17 appsrvr158.keyexample.com sudo[31341]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/touch /home/daved/.ssh
Aug 11 11:58:17 appsrvr158.keyexample.com sudo[31378]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/chmod 640 /home/daved/
Aug 11 11:58:18 appsrvr158.keyexample.com sudo[31411]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/chown daved: /home/dav
Aug 11 11:58:18 appsrvr158.keyexample.com sudo[31445]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/tee -a /home/daved/.ss
Aug 11 11:58:18 appsrvr158.keyexample.com sudo[31444]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/flock /home/daved/.ssh
Aug 11 11:58:19 appsrvr158.keyexample.com sudo[31490]: keyfactor-bash : TTY=unknown ; PWD=/opt/keyfactor-bash-orchestrator ; USER=root ; COMMAND=/bin/test -f /home/daved/.s
```

Figure 599: Status for the Keyfactor Bash Orchestrator Service

For other orchestrators, check to see if the orchestrator service as a whole is running (see details in the specific orchestrator chapters). Start the service if it is not running or restart it if it is running and check again for a heartbeat after a few minutes.

Firewall Ports

At a very basic level, the orchestrator needs to be able to communicate with the Keyfactor Command server(s) on either port 80 or port 443 (depending on the configuration option you've chosen for this connection—see orchestrator specific chapters).

The ports needed for the Keyfactor Universal Orchestrator depend on the functions enabled for the orchestrator. For example, IIS certificate store management uses remote PowerShell (default TCP 5985 and 5986). For SSL discovery and management, any ports configured for scanning need to be open.

The Keyfactor Bash Orchestrator communicates with any remote control targets on port 22 or the alternative port you have configured for SSH. If you are using a non-standard port for SSH, you need to be sure to configure this on both the Keyfactor Command side (see Adding SSH Servers on page 577 in the *Keyfactor Command Reference Guide*) and in the SSH configuration on the orchestrator and remote control targets (sshd_config).

For more information about the firewall ports needed in a Keyfactor Command environment, see Firewall Considerations on page 2765 in the *Keyfactor Command Server Installation Guide*.

Keyfactor Bash Orchestrator Troubleshooting Tips

The Keyfactor Bash Orchestrator has two possible configurations—local and remote. The troubleshooting steps differ depending on whether the server that's not operating as expected is running the orchestrator software (a local installation) or is a control target for the orchestrator (a remote installation). In either case, the best place to start with troubleshooting is in the Keyfactor Command Management Portal to confirm things seem correct on this side of the communication and then configure debug logging on the orchestrator and review those logs.

Successful Inventory and Policy Publishing

In this snippet you see a successful inventory showing keys found for the Linux users ginag and svc_greenchicken and a logon found for the Linux user zadams with no key found. You see that the server is configured in *inventory and publish policy mode*, since after performing the inventory the server went through the steps of publishing logons and keys. Details about these are not written to the log.

```
Tue Aug 11 18:07:45 UTC 2020 [Debug]: Sending request to 'https://key-
factor.keyexample.com/KeyfactorAgents/SshSync/1/Configure' with payload '{"SessionToken":
"5451f7aa-4fd5-4bf5-a563-2e4f7bd3ed3f", "JobId": "b835bde8-8174-447a-b351-810e582148c0"}'
Tue Aug 11 18:07:45 UTC 2020 [Debug]: Configure Response for job with id 'b835bde8-8174-447a-b351-
810e582148c0': {"Host-
name":"appsrvr79.keyexample.com","InventoryCompleteEndpoint":"/SshSync/1/InventoryComplete",
"Port":22,"AuditId":7642,"JobCancelled":false,"Result":{"Status":1,"Error":null}}
Tue Aug 11 18:07:46 UTC 2020 [Debug]: Using sshd_config file '/etc/ssh/sshd_config' on server
'appsrvr79.keyexample.com' for job with id 'b835bde8-8174-447a-b351-810e582148c0'
Tue Aug 11 18:07:46 UTC 2020 [Info]: Beginning local inventory job on server 'appsr-
vr79.keyexample.com' for job with id 'b835bde8-8174-447a-b351-810e582148c0'
Tue Aug 11 18:07:49 UTC 2020 [Debug]: Sending request to 'https://key-
factor.keyexample.com/KeyfactorAgents/SshSync/1/InventoryComplete' with payload '{"Status":2,"Res-
ults": [{
"user": "ginag",
"lastlogon": "",
"keys": [ "ssh-rsa AAAAB3NzaC1yc2EAAAA[truncated for display purposes]9M5vl6f Gina G. Gant" ]
},{
"user": "zadams",
"lastlogon": "",
"keys": []
},{
"user": "svc_greenchicken",
"lastlogon": "",
"keys": [ "ssh-rsa AAAAB3NzaC1yc2EAAAAD[truncated for display purposes]vicWhZOd John W. Smith" ]
}],"SessionToken": "5451f7aa-4fd5-4bf5-a563-2e4f7bd3ed3f","JobId": "b835bde8-8174-447a-b351-
810e582148c0"}'
Tue Aug 11 18:07:49 UTC 2020 [Debug]: Inventory Complete Response for job with id 'b835bde8-8174-
447a-b351-810e582148c0' on server 'appsrvr79.keyexample.com': {"SshDesiredState":[{"User-
name":"ginag","Keys":["ssh-rsa AAAAB3NzaC1yc2EAAAA[truncated for display purposes]9M5vl6f Gina G.
Gant"]},{"Username":"zadams","Keys":[]},{"Username":"svc_greenchicken","Keys":["ssh-rsa AAAAB3Nz-
aC1yc2EAAAAD[truncated for display purposes]vicWhZOd John W. Smith"]}],"Result":{"Status":1,"Er-
ror":null}}
Tue Aug 11 18:07:49 UTC 2020 [Info]: Enforcing publish policy on server 'appsrvr79.keyexample.com'
for job with id 'b835bde8-8174-447a-b351-810e582148c0'
Tue Aug 11 18:07:52 UTC 2020 [Info]: Publishing logons on local server 'appsrvr79.keyexample.com'
```

```
for job with id 'b835bde8-8174-447a-b351-810e582148c0'
Tue Aug 11 18:07:52 UTC 2020 [Info]: Published logons successfully on server 'appsr-
vr79.keyexample.com' for job 'b835bde8-8174-447a-b351-810e582148c0'
Tue Aug 11 18:07:52 UTC 2020 [Info]: Publishing keys on local server 'appsrvr79.keyexample.com' for
job with id 'b835bde8-8174-447a-b351-810e582148c0'
Tue Aug 11 18:07:54 UTC 2020 [Info]: Published keys successfully on server 'appsr-
vr79.keyexample.com' for job 'b835bde8-8174-447a-b351-810e582148c0'
Tue Aug 11 18:07:54 UTC 2020 [Debug]: Sending request to 'https://key-
factor.keyexample.com/KeyfactorAgents/SshSync/1/Complete' with payload '{"SessionToken": "5451f7aa-
4fd5-4bf5-a563-2e4f7bd3ed3f", "JobId": "b835bde8-8174-447a-b351-810e582148c0", "Status": 2}'
Tue Aug 11 18:07:54 UTC 2020 [Info]: Execution of 'b835bde8-8174-447a-b351-810e582148c0' on server
'appsrvr79.keyexample.com' complete.
```

Validate Service Account Logon

During installation of the orchestrator, a local Linux user account should be created automatically as an identity under which the orchestrator service will operate. This allows the orchestrator to run as a non-root user. On servers on which you install the orchestrator directly, the following Linux user account is created:

keyfactor-bash

On servers configured as remote control targets, the following Linux user account is created:

keyfactor-bash-orchestrator-svc

You can validate that the user has been created and has the correct configuration be reviewing the /etc/passwd file.

In a command shell, output the content of the /etc/passwd file to the screen:

```
cat /etc/passwd
```

In the output from this command, look for the entry for the keyfactor-bash or keyfactor-bash-orchestrator-svc user. It will look similar to one of these:

```
keyfactor-bash:x:978:976::/home/keyfactor-bash:/bin/bash
keyfactor-bash-orchestrator-svc:x:112:65534::/opt/keyfactor-bash-orchestrator-
client:/bin/bash
```

On the remote control target server, you should find an entry in the sshd_config file that directs the service account logon over to the install path for the client to find the authorized_keys file for the service account user, like so:

```
Match User keyfactor-bash-orchestrator-svc
AuthorizedKeysFile    /opt/keyfactor-bash-orchestrator-client/authorized_keys
```

On both the orchestrator and remote control target servers, you should find a file in the /etc/sudoer.d directory named for the service name of the orchestrator or remote control target user (keyfactor-bash or keyfactor-bash-orchestrator-svc) and containing a list of commands the orchestrator is allowed to execute as root. For example:

```
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /bin/ls
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /bin/cat
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /usr/bin/test
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /bin/rm
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /usr/bin/tee
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /bin/touch
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /bin/chmod
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /bin/chown
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /usr/bin/gpasswd
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /usr/sbin/usermod
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /bin/sed
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /usr/bin/flock
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /bin/mkdir
keyfactor-bash appsrvr79.keyexample.com = (root) NOPASSWD: /usr/sbin/adduser
```

Validate Remote Control Target Public Key

The orchestrator connects to the remote control targets it is managing using SSH with a public key pair. On the orchestrator, the key pair is stored in the .ssh directory under the directory where the orchestrator is installed. By default, this is:

```
/opt/keyfactor-bash-orchestrator/.ssh
```

Both the private key (id_rsa) and public key (id_rsa.pub) are found here.

In a command shell, output the content of the public key file to the screen:

```
cat id_rsa.pub
```

On the remote control target, the public key of the key pair is stored in the authorized_keys file for the remote control target service account, which is found in the remote control install path. By default, this is:

```
/opt/keyfactor-bach-orchestrator-client
```

In a command shell, output the content of the authorized_keys file to the screen:

```
cat authorized_keys
```

Compare the public key string from the remote control target authorized_keys file to the public key string from the orchestrator id_rsa.pub file. They should match exactly. If they do not match, the remote control target is not using the correct public key, which will cause connection attempts made to it from the orchestrator to fail.

Tip: You should also see in the .ssh directory on the orchestrator a file named by hostname (e.g. appsrvr80.keyexample.com) for each of the remote control targets managed by the orchestrator. These contain a list of known, trusted host key stores. If this file has not be created for your remote control target, connectivity to the target is failing at a very fundamental level (before the stage of a public key mismatch). See Firewall Ports on page 3010.

Keyfactor Bash Orchestrator Log Messages

If the orchestrator is managing more than one server (remote control targets), it can be difficult to interpret the logs, because the orchestrator operates in a multi-threaded manner and log messages for jobs with different servers will be mixed together. Find a message related to the job you're interested in and look for the ID for that job. Then look for all other messages referencing that ID.

Look for error messages in the log. These should appear with the word *Error* in brackets just after the date like so:

```
Tue Aug 11 19:14:33 UTC 2020 [Error]: Error occurred during job with id 'b835bde8-8174-447a-b351-810e582148c0' on server 'appsrvr79.keyexample.com': An error occurred attempting to configure the job 'b835bde8-8174-447a-b351-810e582148c0'
```

This particular message doesn't tell you very much except that this job was unable to complete for some reason. If you look at the debug messages that appear immediately before and after the error message, they may provide more information.

This message indicates that the orchestrator was unable to make an SSH connection to the remote control target named in the message:

```
Mon Aug 10 23:36:10 UTC 2020 [Error]: Error occurred during job with id '3f04f552-05fd-4c90-b3b1-edeec70878bb' on server 'appsrvr80.ubuntu.keyexample.com': Unable to connect to 'appsr-vr80.ubuntu.keyexample.com' on port '22' via SSH
```

This could happen for a number of reasons. Perhaps the hostname configured for the remote target is incorrect. Perhaps the public key on the remote target is incorrect. It can be helpful in this case to check the Linux syslog on the orchestrator for more context on the message. For example, this set of messages from the Linux syslog reveals that the public key on the target is invalid in some fashion:

```
Aug 11 13:03:04 appsrvr158 keyfactor-bash[29417]: Testing 'keyfactor-bash-orchestrator-svc' on
server 'appsrvr80.keyexample.com' via SSH for job with id 'eeabd541-b9d2-46d2-a215-9cb99fed4adc'...
Aug 11 13:03:04 appsrvr158 keyfactor-bash-orchestrator.sh[932]: keyfactor-bash-orchestrator-
svc@appsrvr80.keyexample.com: Permission denied (publickey).
Aug 11 13:03:30 appsrvr158 keyfactor-bash[29486]: Error occurred during job with id 'eeabd541-b9d2-
46d2-a215-9cb99fed4adc' on server 'appsrvr80.keyexample.com': Unable to connect to 'appsr-
vr80.keyexample.com' on port '22' via SSH
```

For information on troubleshooting public key issues with remote control targets, see Validate Remote Control Target Public Key on page 3013. For more information on troubleshooting remote control target issues in general, see Remote Control Target Logs below. For information on what successful inventory and publish policy log messages look like, see Successful Inventory and Policy Publishing on page 3011.

Remote Control Target Logs

Unlike on the orchestrator itself, where you can enable debug logging to see a more detailed picture of what's going on when the orchestrator attempt to connect or run a job, on a remote control target, the only logs available are the SSH logs showing attempts by the orchestrator to make a remote connection into the target and then the commands the orchestrator runs from an SSH perspective. These logs are found in the Linux system log where SSH logs are consolidated. The name and location of this will vary by operating system, but it is often found in /var/log by default (*auth.log* or *secure* is common). A large number of entries are generated in the log on a successful connection for inventory or inventory and policy publishing, so it can be difficult to interpret the logs.

In these logs you can check to see if the orchestrator is successfully making an SSH connection. If it isn't, you may see some messages that will help determine why it isn't. If it's successfully making the initial connection but then failing further along in the process, this log may also help reveal that. Perhaps one of the commands that the service account needs to run isn't in the expected path, for example.

When the orchestrator first connects to the remote control target, the log entries on the target will look something like:

```
Aug 11 17:36:51 appsrvr80 sshd[95543]: Accepted publickey for keyfactor-bash-orchestrator-svc from
10.4.3.158 port 47778 ssh2: RSA SHA256:u5zNB4UEoPNcax5p4fBbkkWaoiWq6AcEkA65XdzUkM4
Aug 11 17:36:51 appsrvr80 sshd[95543]: pam_unix(sshd:session): session opened for user keyfactor-
bash-orchestrator-svc by (uid=0)
Aug 11 17:36:51 appsrvr80 systemd-logind[656]: New session 13019 of user keyfactor-bash-orches-
trator-svc.
Aug 11 17:36:51 appsrvr80 systemd: pam_unix(systemd-user:session): session opened for user
keyfactor-bash-orchestrator-svc by (uid=0)
```

```
Aug 11 17:36:51 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/bin/cat /etc/ssh/sshd_config
```

An inventory of an authorized_keys file for a user will appear as a series of entries, something like:

```
Aug 11 18:11:28 appsrvr164 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=-
=/opt/keyfactor-bash-orchestrator-client ; USER=root ; COMMAND=/bin/test -f /home/j-
smith/.ssh/authorized_keys
Aug 11 18:11:28 appsrvr164 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=-
=/opt/keyfactor-bash-orchestrator-client ; USER=root ; COMMAND=/bin/ls -l /home/j-
smith/.ssh/authorized_keys
Aug 11 18:11:28 appsrvr164 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=-
=/opt/keyfactor-bash-orchestrator-client ; USER=root ; COMMAND=/bin/cat /home/j-
smith/.ssh/authorized_keys
```

Removal of a rogue key on a remote control target under management (in *inventory and publish policy* mode) will appear as a series of entries where the authorized_keys file is removed, recreated and repopulated with any valid keys (none in this case), like:

```
Aug 12 09:01:24 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/usr/bin/test -f /home/jsmith/.ssh/authorized_keys
Aug 12 09:01:24 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/bin/rm /home/jsmith/.ssh/authorized_keys
Aug 12 09:01:25 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/usr/bin/test -f /home/jsmith/.ssh/authorized_keys
Aug 12 09:01:25 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/usr/bin/test -d /home/jsmith/.ssh
Aug 12 09:01:25 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/usr/bin/touch /home/jsmith/.ssh/authorized_keys
Aug 12 09:01:25 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/bin/chmod 640 /home/jsmith/.ssh/authorized_keys
Aug 12 09:01:25 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/bin/chown jsmith: /home/jsmith/.ssh/authorized_keys
Aug 12 09:01:25 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/usr/bin/flock /home/jsmith/.ssh/authorized_keys
echo
Aug 12 09:01:25 appsrvr80 sudo: keyfactor-bash-orchestrator-svc : TTY=unknown ; PWD=/opt/keyfactor-
bash-orchestrator-client ; USER=root ; COMMAND=/usr/bin/tee -a /home/jsmith/.ssh/authorized_keys
```

General Errors

Below are some possible errors you might encountered and some suggested troubleshooting tips or solutions.

Unable to connect to the remote server

Here is an example of some very similar errors you might see when trying to connect to a target machine to inventory a certificate store or execute a management or discovery job on a certificate store:

```
Error: Unable to connect to the remote server - No connection could be made because
the target machine actively refused it 192.196.12.12:443 (80131500)
```

```
Error: Unable to complete the inventory operation. One or more errors occurred.
An error occurred while sending the request.
Unable to connect to the remote server
A connection attempt failed because the connected party did not properly respond after
a period of time, or established connection failed because connected host has failed
to respond 192.168.12.12:443 (80131500)
```

```
Error: Unable to connect to the remote server (80131509)
```

```
Error occurred during job with id 'b5e93ae6-df3b-4b36-9640-b41146db6d36' on server
'appsrvr13.keyexample.com': Unable to connect to 'appsrvr13.keyexample.com' on port
'22' via SSH
```

Messages of this type are generally the result of the target server being inaccessible. This might happen if the server was turned off or in maintenance mode. Perhaps there is a network problem routing to that server. If the certificate store has never worked in Keyfactor Command, perhaps there is a typo in the server name configuration.

Request Entity Too Large

You may encounter this error when doing an inventory of an IIS certificate store:

```
Error: Response status code does not indicate success: 413 (Request Entity Too Large).
(80131500)
```

This is an indication that the certificate store you are inventorying contains more certificates (or more precisely, the certificates add up to a total number of bytes greater) than IIS on the Keyfactor Command server is configured to accept. To resolve this, adjust the values on the IIS server that control the upload limits. For example, the *maxAllowedContentLength*. See Monitoring Network Scan Jobs with View Scan Details on page 465 in the *Keyfactor Command Reference Guide*) on fine tuning SSL monitoring for more information.

IIS Error 403.16

You may receive a 403.16 error while trying to authenticate an orchestrator to Keyfactor Command using certificate authentication. On the face of it, this error indicates that the chain for the certificate you're using to authenticate is not trusted by the Keyfactor Command server. First,

check to be sure that your certificate is trusted by the Keyfactor Command server. But if your certificate is fully trusted and you're still getting this error, what then?

This error can indicate that the trusted root store on the Keyfactor Command server contains a certificate that is not a root certificate (for example, an intermediate certificate is accidentally in the root store). To check this, open the Local Computer certificates MMC on the Keyfactor Command server, drill down to Certificates under the Trusted Root Certificate Authorities and scan for any certificates where the *Issued To* does not match the *Issued By*. Remove any certificates you find like this.

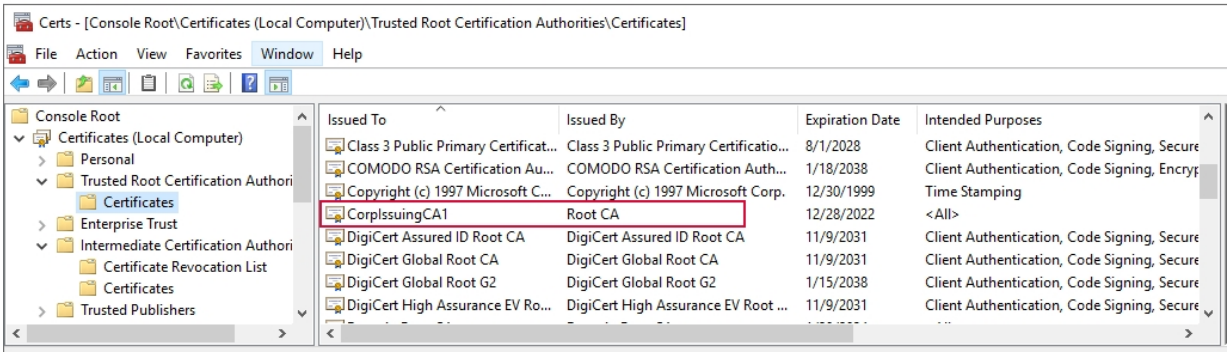


Figure 600: Certificate Incorrectly in the Trusted Root Certificate Store

Error: An attempt was made to load a program with an incorrect format.

If you receive an error similar to the following:

```
Could not load file or assembly 'Keyfactor.CAClient.Microsoft.DCOM, Version=2.1.1.0,
Culture=neutral, PublicKeyToken=0ed89d330114ab09'. An attempt was made to load a
program with an incorrect format.
```

This may indicate that the Keyfactor Universal Orchestrator was installed without the Microsoft Visual C++ Redistributable x64 required to manage certificates from remote Microsoft CAs (see System Requirements on page 2880).

Error: The remote certificate is invalid because of errors in the certificate chain

If you receive an error similar to the following (some portions of message removed for clarity):

```
2023-02-15 11:54:27.6600   Keyfactor.Orchestrators.JobEngine.SessionJobExecutor [Error] - Error in
SessionManager: Unable to register session.

The SSL connection could not be established, see inner exception.
```

```
The remote certificate is invalid because of errors in the certificate chain: Revoc-
ationStatusUnknown, OfflineRevocation
```

This may indicate that the Keyfactor Universal Orchestrator cannot access the CRL(s) for the SSL certificate used to secure the Keyfactor Command server (see System Requirements on page 2880).

To check this:

1. Enable at least debug level logging (see Configure Logging for the Universal Orchestrator on page 2950).
2. Either wait for the orchestrator to attempt to register again, or restart the orchestrator service (see Start the Universal Orchestrator Service on page 2953) to force an immediate attempt to register.
3. Look in the logs for a log message similar to the following (referencing your Keyfactor Command server name):

   ```
   2023-02-15 12:08:14.6076 Keyfactor.Orchestrators.Core.Http.KeyfactorHttpClient
   [Debug] - Sending request to
   'https://keyfactor.keyexample.com/KeyfactorAgents/Session/Register'
   ```

4. Visit the referenced URL (https://key-factor.keyexample.com/KeyfactorAgents/Session/Register) in a browser on the orchestrator server. This should give you a response of:

   ```
   The requested resource does not support http method 'GET'.
   ```

5. In the browser, view details for the certificate (the exact method for this will vary depending on the browser) and check the *CRL Distribution Points* field in the certificate.

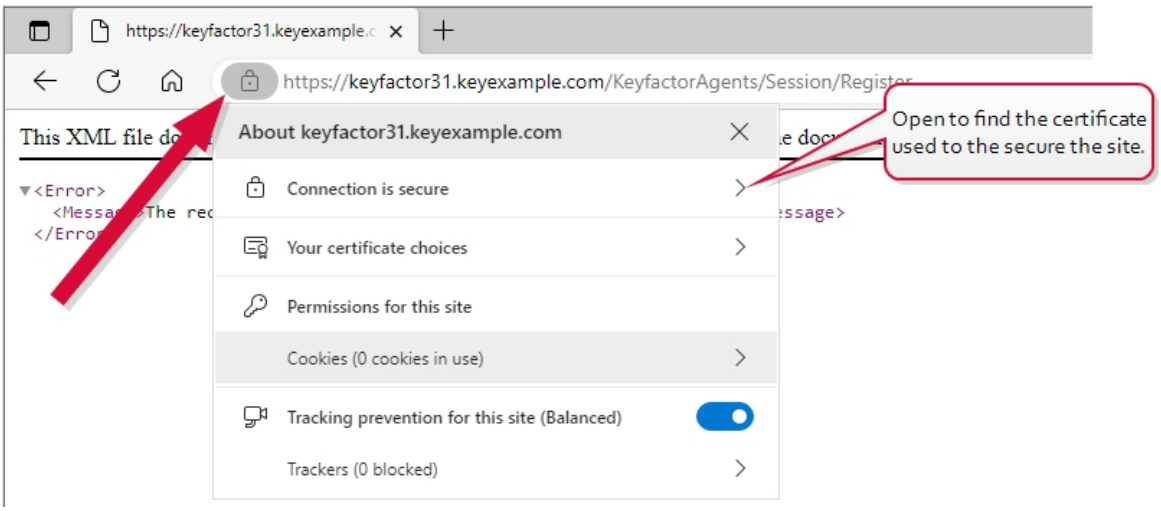


Figure 601: Find the Certificate for the Keyfactor Command Web Site

6. In the same browser on the orchestrator server, attempt to browse to the URL for the CRL (assuming it's a URL).

7. If the CRL downloads without error, then likely CRL access is not the issue. Open the CRL and check the *Next update* date to see if it's in the past (indicating the CRL is out of date).

Note: CRL checks are done on port 80 since the CRL lookup is part of the validation of the server's SSL certificate. This means the CRLs need to be available at an http URL. The CRL file that is retrieved is signed by the CA, so although the network communication is not encrypted when retrieving it, the data that is being validated can't be tampered with (because it is signed).

Tip: The Keyfactor Universal Orchestrator can be installed without checking the CRL of the Keyfactor Command if desired. Use the *-NoRevocationCheck* option for the Windows orchestrator (see -NoRevocationCheck on page 2906), the *--no-revocation-check* option for the Linux orchestrator (see --no-revocation-check on page 2919), or the *AppSettings__CheckServerCertificateRevocation* option for the orchestrator in Linux containers (see Table 854: Linux Container Parameters).

Remote Management Helpful Tools

The following tips are useful for servers being remotely managed using PowerShell remoting and WinRM.

- Test the connection from the orchestrator server to the remotely managed Windows server:

```
Test-netConnection -ComputerName <target> -Port 5986 or Test-netConnection -
ComputerName <target> -Port 5985
```

- Test PS Session from the orchestrator server to the remotely managed server:

  ```
  Enter-PSSEssion -ComputerName <target>
  ```

- On the remotely managed server, check what's available:

  ```
  winrm enumerate winrm/config/listener
  ```

- Enable secure winrm:

  ```
  winrm quickconfig -transport:https
  ```

- Check the secure winrm port certificate:

  ```
  gci -path cert:\localmachine\my |ft -property thumbprint,subject,NotBefore,NotAfter
  ```

5.6 Appendices

- Appendix - Generate New Credentials for the Java Agent below
- Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC on page 3023
- Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication with Certificates Stored in Active Directory on page 3035
- Appendix - Set up the Universal Orchestrator to Use a Forwarding Proxy on page 3049

5.6.1 Appendix - Generate New Credentials for the Java Agent

Under some circumstances, you may find it necessary to generate new credentials for the Java Agent. This can happen, for example, if you make a change to the hostname of the machine on which the Java Agent is running. The credentials file stores the username and password for the service account user that allows the Java Agent to communicate with Keyfactor Command—the identity for the agent (see Create Service Accounts for the Java Agent on page 2969)—encrypted with the hostname to prevent the file from being used on machines other than the machine on which the agent has been installed.

Log messages that indicate a new credentials file is needed look similar to the following:

```
2020-10-02 15:21:43.307 [Scheduler_Worker-1] ERROR com.css_security.cms.apache.http.HttpClientFactory
- Given final block not properly padded. Such issues can arise if a bad key is used during decryp-
tion.
2020-10-02 15:21:43.307 [Scheduler_Worker-1] ERROR com.css_security.cms.apache.http.HttpClientFactory
- Could not decrypt credentials file at config\install.creds
2020-10-02 15:21:43.526 [Scheduler_Worker-1] INFO  com.css_security.cms.apache.http.HttpClientFactory
- Your machine key may have changed. Reencrypt credentials using local machine key.
```

```
2020-10-02 15:21:43.541 [Scheduler_Worker-1] INFO  com.css_security.cms.apache.http.HttpClientFactory
- Generate new credentials by running included cms-credential-encryptor utility
```

To generate a new credentials file on Windows:

1. Open a command prompt using the “Run as administrator” option.
2. Change directories to the directory in which the Java Agent is installed. By default, this is:

   C:\Program Files\Keyfactor\Keyfactor Java Agent
3. Type the following command to generate a new credentials file in the current directory:

   ```
   java -jar CSS.CMS.CredentialEncryptor.jar encode-basic install.creds
   ```
4. Locate the existing credentials file in the config directory under the installed directory. By default, this is:

   C:\Program Files\Keyfactor\Keyfactor Java Agent\config
5. Delete or name off the existing install.creds file in the config directory and copy the new install.creds file from the base install directory to the config directory.
6. Restart the Java Agent service (see Start the Keyfactor Java Agent Service on page 2989).
7. Review the log messages to confirm that credential errors are no longer occurring (see Configure Logging for the Java Agent on page 2986).

To generate a new credentials file on Linux:

1. Open a command shell.
2. Change directories to the directory in which the Java Agent is installed. By default, this is:

   /opt/keyfactor-java-agent
3. As a user with rights to write to the current directory (or use sudo), type the following command to generate a new credentials file in the current directory:

   ```
   java -jar CSS.CMS.CredentialEncryptor.jar encode-basic install.creds
   ```
4. Locate the existing credentials file in the config directory under the installed directory. By default, this is:

   /opt/keyfactor-java-agent/config
5. Delete or name off the existing install.creds file in the config directory and copy the new install.creds file from the base install directory to the config directory.
6. Restart the Java Agent service (see Start the Keyfactor Java Agent Service on page 2989).
7. Review the log messages to confirm that credential errors are no longer occurring (see Configure Logging for the Java Agent on page 2986).

5.6.2 Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC

The Keyfactor Universal Orchestrator can be configured to support TLS termination at a reverse proxy or network edge device such as a Citrix ADC (a.k.a. NetScaler) or F5. The orchestrator supports using either basic authentication or client certificate authentication between the orchestrator and the Keyfactor Command orchestrator endpoint. When a client certificate is used for the segment between the orchestrator and the reverse proxy, the reverse proxy authenticates the orchestrator with the provided client certificate and then sends the certificate on to Keyfactor Command as an added request header to authenticate the orchestrator to Keyfactor Command with the original certificate. The orchestrator is authenticated and authorized to make the connection to Keyfactor Command in one of two ways:

- A username and password with appropriate permissions within Keyfactor Command are stored on the reverse proxy and presented to Keyfactor Command as part of the request. Basic authentication is used to authenticate the reverse proxy to IIS on the Keyfactor Command server. The same credentials provide authorization for the orchestrator in Keyfactor Command. The original certificate from the orchestrator, provided in a request header, authenticates the orchestrator to the Keyfactor Command orchestrator endpoint.
- A username and password with appropriate permissions within Keyfactor Command are stored in IIS on the Keyfactor Command. In this scenario, a second client certificate residing on the reverse proxy is used to authenticate the reverse proxy to IIS on the Keyfactor Command server. The basic authentication credentials provide authorization for the orchestrator in Keyfactor Command and the original client certificate from the request header provides authentication. The basic authentication credentials are stored locally and do not need to travel over the network. The original certificate from the orchestrator, provided in a request header, authenticates the orchestrator to the Keyfactor Command orchestrator endpoint.

The following instructions cover one method of configuring a Citrix ADC device to support these.

> **Tip:** The following provides instructions for using the Citrix ADC GUI interface to create the appropriate configuration. The same configuration could be accomplished using the command line interface.

Complete the following steps and then configure the orchestrator to enable client certificate authentication as per the installation instructions (see --client-auth-certificate (Client Certificate Authentication) on page 2916 or Install the Universal Orchestrator on Windows on page 2898).

Define Rewrite Actions in Citrix

Create the following two rewrite actions.

> **Tip:** If you're using a second client certificate to authenticate the proxy to Keyfactor Command, you only need to create the first of these actions.

Capture the client certificate from the orchestrator:

1. In the Citrix ADC GUI, browse to *AppExpert > Rewrite > Actions*.
2. On the Rewrite Actions page, click **Add**.
3. On the Create Rewrite Action page, enter a **Name** for the action that will take the certificate received from the orchestrator and convert it to PEM format (e.g. CaptureClientCert).
4. Give the action a *Type* of *INSERT_HTTP_HEADER*.
5. Give the action a *Header Name* (e.g. NS-ClientCert). Be sure to make a note of this header name. You will need it later when you configure certificate authentication for the orchestrator.
6. Enter an *Expression* to convert the client authentication certificate to PEM format:

   ```
   CLIENT.SSL.CLIENT_CERT.TO_PEM
   ```

7. Enter *Comments* if desired and click **OK** to save the action.

Store basic authentication credentials to authenticate the proxy to IIS on the Keyfactor Command server and provide authorization information:

1. Click **Add** to add another action.
2. On the Create Rewrite Action page, enter a **Name** for the action that will send the basic authentication credentials for the orchestrator to Keyfactor Command (e.g. SendServiceCreds).
3. Give the action a *Type* of *INSERT_HTTP_HEADER*.
4. Give the action a *Header Name* of *Authorization*.
5. Enter an *Expression* to send Base64-encoded basic authentication credentials to the Keyfactor Command server (where *service@keyexample.com* and *MySecurePassword* are the correct service name and password for your environment):

   ```
   "Basic "+("service@keyexample.com"+":"+"MySecurePassword").B64ENCODEM
   ```

6. Enter *Comments* if desired and click **OK** to save the action.

Define Rewrite Policies in Citrix

Create the following two rewrite policies.

> **Tip:** If you're using a second client certificate to authenticate the proxy to Keyfactor Command, you only need to create the first of these policies.

Put the client certificate from the orchestrator in the header:

1. In the Citrix ADC GUI, browse to *AppExpert > Rewrite > Policies*.
2. On the Rewrite Policies page, click **Add**.

3. On the Create Rewrite Policy page, enter a **Name** for the policy that will confirm that a certificate has been received from the orchestrator and run the action to convert it to PEM format (e.g. NS-GetCert).
4. Give the policy the **Action** you created in the previous section to capture the client authentication certificate (e.g. CaptureClientCert).
5. Define a **Log Action** if desired.
6. Set the **Undefined-Result Action** to *-Global-undefined-result-action-*.
7. Enter an *Expression* to validate that the client authentication certificate has been received from the orchestrator:

   ```
   CLIENT.SSL.CLIENT_CERT.EXISTS
   ```

8. Enter *Comments* if desired and click **OK** to save the policy.

Send the basic authentication credentials to the Keyfactor Command server:

1. Click **Add** to add another policy.
2. On the Create Rewrite Policy page, enter a **Name** for the policy that will send the basic authentication credentials for the orchestrator to the Keyfactor Command server (e.g. NS-SendCreds).
3. Give the policy the **Action** you created in the previous section to send the basic authentication credentials (e.g. SendServiceCreds).
4. Define a **Log Action** if desired.
5. Set the **Undefined-Result Action** to *-Global-undefined-result-action-*.
6. Enter an *Expression* to confirm that the authorization header does not already exist in the request header:

   ```
   HTTP.REQ.HEADER("Authorization").EXISTS.NOT
   ```

7. Enter *Comments* if desired and click **OK** to save the policy.

Define a Responder Policy in Citrix

Create the following responder policy.

Validate that the client certificate presented by the orchestrator was issued by the specified issuing CA:

1. In the Citrix ADC GUI, browse to *AppExpert > Responder > Policies*.
2. On the Responder Policies page, click **Add**.

3. On the Create Responder Policy page, enter a **Name** for the policy that will validate that the certificate received from the orchestrator was issued by the correct CA (e.g. NS-ValidateIssuer).
4. Select an **Action** of *Reset*.
5. Define a **Log Action** if desired.
6. Do not configure an **AppFlow Action**.
7. Set the **Undefined-Result Action** to *-Global-undefined-result-action-*.
8. Enter an *Expression* to confirm that the certificate received from the orchestrator was issued from the correct issuing CA (where *CorpIssuingCA* is the logical name of your CA):

   ```
   CLIENT.SSL.CLIENT_CERT.ISSUER.CONTAINS("CorpIssuingCA").NOT
   ```

   Tip: Connections from the orchestrator will fail if the client authentication certificate was issued by any CA other than the one configured here. You can use AND logic to add more than one CA. For example:

   ```
   CLIENT.SSL.CLIENT_CERT.ISSUER.CONTAINS("CorpIssuingCA1").NOT &&
   CLIENT.SSL.CLIENT_CERT.ISSUER.CONTAINS("CorpIssuingCA2").NOT
   ```

   With this expression, certificates issued from either one of these CAs would be accepted.
9. Enter *Comments* if desired and click **OK** to save the policy.

Update the Virtual Server in Citrix

Important: Once you modify the virtual server to require certificates for authentication, many other Keyfactor Command transactions will no longer function if they are sharing the same virtual server. Be sure that you are using a separate virtual server for incoming requests to /KeyfactorAgents on the Keyfactor Command server versus other types of requests. The following instructions refer to setting all policies on a single load balancing virtual server, but your configuration may include multiple virtual servers of other types, which may require slight modifications to these instructions.

Modify the configuration for your load balancing virtual server that is used for Keyfactor Command KeyfactorAgent requests as follows.

Configure the Citrix device to authenticate the orchestrator using its client certificate:

1. In the Citrix ADC GUI, browse to *Traffic Management > Load Balancing > Virtual Servers*.
2. On the Virtual Servers page, select your virtual server and click **Edit**.
3. In the SSL Parameters section, click to edit, check the **Client Authentication** box, and set the **Client Certificate** dropdown to **Mandatory**.

Associate the two rewrite policies.

Tip: If you're using a second client certificate to authenticate the proxy to Keyfactor Command, you only need to associate the first of these policies.

Configure the policy to include the certificate in the header:

1. On the Virtual Servers page, under Advanced Settings expand Policies.
2. In the Policies section, click the plus to add a new policy.
3. On the Choose Type page, select **Choose Policy***Rewrite* and **Choose Type***Request* and click **Continue**.
4. On the Choose Type page, click **Add Binding**.
5. On the Policy Binding page, click the **Select Policy** field and on the Rewrite Policies page select the radio button for the rewrite policy you created to capture the client authentication certificate (e.g. NS-GetCert). Click **Select** to save the selection.
6. On the Policy Binding page, set a **Priority** of 110.
7. Set **Goto Expression** to *Next*.
8. Set **Invoke LabelType** to *None*.
9. Click **Bind** to save the binding.

Configure the policy to send the basic authentication credentials to the Keyfactor Command server:

1. On the Choose Type page for Rewrite Request, click **Add Binding**.
2. On the Policy Binding page, click the **Select Policy** field and on the Rewrite Policies page select the radio button for the rewrite policy you created to send the service account credentials to the Keyfactor Command server (e.g. NS-SendCreds). Click **Select** to save the selection.
3. On the Policy Binding page, set a **Priority** of 120.
4. Set **Goto Expression** to *Next*.
5. Set **Invoke LabelType** to *None*.
6. Click **Bind** to save the binding.
7. Click **Close** to return to the virtual server settings page.

Associate the responder policy:

1. On the Virtual Servers page, in the Policies section, click the plus to add a new policy.
2. On the Choose Type page, select **Choose Policy** *Responder* and **Choose Type** *Request* and click **Continue**.
3. On the Choose Type page, click **Add Binding**.
4. On the Policy Binding page, click the **Select Policy** field and on the Responder Policies page select the radio button for the responder policy you created to validate the issuer of the client authentication certificate (e.g. NS-ValidateIssuer). Click **Select** to save the selection.
5. On the Policy Binding page, set a **Priority** of 100.
6. Set **Goto Expression** to *END*.
7. Set **Invoke LabelType** to *None*.
8. Click **Bind** to save the binding.
9. Click **Close** to return to the virtual server settings page.

Configure Keyfactor Command for Client Certificate Authentication

Once you have all the components configured on Citrix, you're ready to configure Keyfactor Command to enable client certificate authentication for the orchestrators. Once you do this, all orchestrators connecting to this instance of Keyfactor Command will be required to provide a certificate to authenticate. If you have some orchestrators deployed that do not support certificate authentication (e.g. Java agents), you will need to design a solution with multiple Keyfactor Command servers to support multiple authentication types. Contact your Keyfactor representative for assistance with this.

To configure Keyfactor Command to require client certificate authentication for orchestrators:

1. On the Keyfactor Command server, open the Keyfactor Configuration Wizard.
2. In the Certificate Authentication section of the Orchestrators tab, check the **Enabled** box.
3. In the **Certificate Authentication HTTP Header** field, enter the *Header Name* you gave to the rewrite action you created to capture the certificate from the orchestrator (e.g. NS-ClientCert). Keyfactor Command uses the certificate supplied in this header to identify the orchestrator attempting to authenticate.
4. In the **Certificate Authentication Username** and **Certificate Authentication Password** fields, enter the credentials for an Active Directory service account for the orchestrator(s).

   Tip: The service account entered here does not need to match the service account entered on the Citrix device to authenticate the orchestrator.

5. Click **Verify Configuration** and **Apply Configuration**.

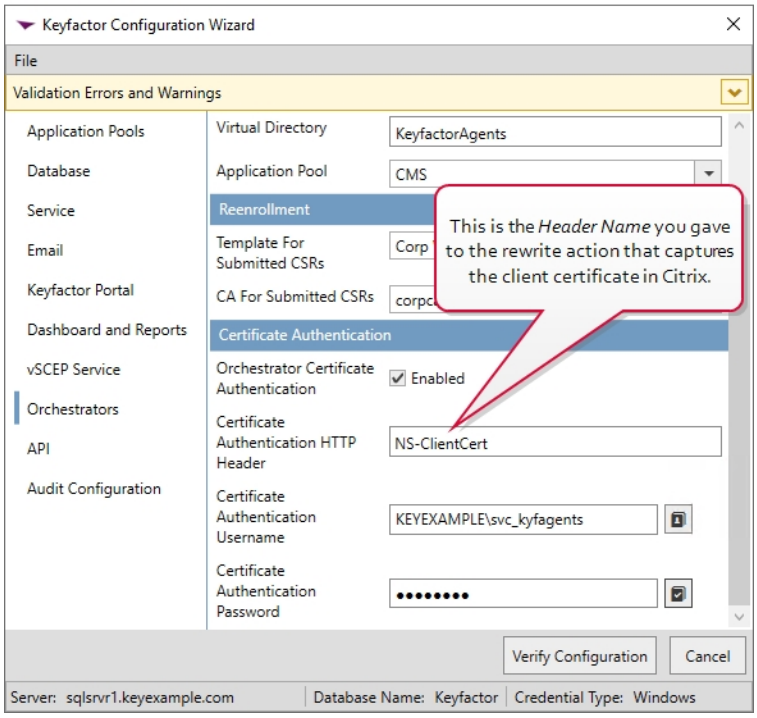


Figure 602: Configure Keyfactor Command for Client Certificate Authentication

Configure IIS to Provide Credentials When a Second Client Certificate is Used to Authenticate the Proxy

If you have opted to configure the Citrix ADC device to use a client certificate to authenticate from the device to the Keyfactor Command server instead of submitting basic authentication credentials from the device, you will need to configure IIS on the Keyfactor Command server to recognize the client certificate for authentication and then use basic authentication credentials on the Keyfactor Command server to provide authorization to Keyfactor Command. In addition, you will need to configure Keyfactor Command to force it to use the client certificate from the orchestrator stored in the header to authenticate the orchestrator, not the client certificate presented by the proxy in the second hop of the transaction.

Install the Required Windows Module

On your Keyfactor Command server, install the following additional module:

- *IIS Client Certificate Mapping Authentication* (rather than *Client Certificate Mapping Authentication*)

> **Tip:** It's fine to install both *IIS Client Certificate Mapping Authentication* and *Client Certificate Mapping Authentication*, but the former is what's needed for this solution.

If you have more than one Keyfactor Command server with separated roles, this only needs to be installed on the server accepting traffic to the /KeyfactorAgents web application.

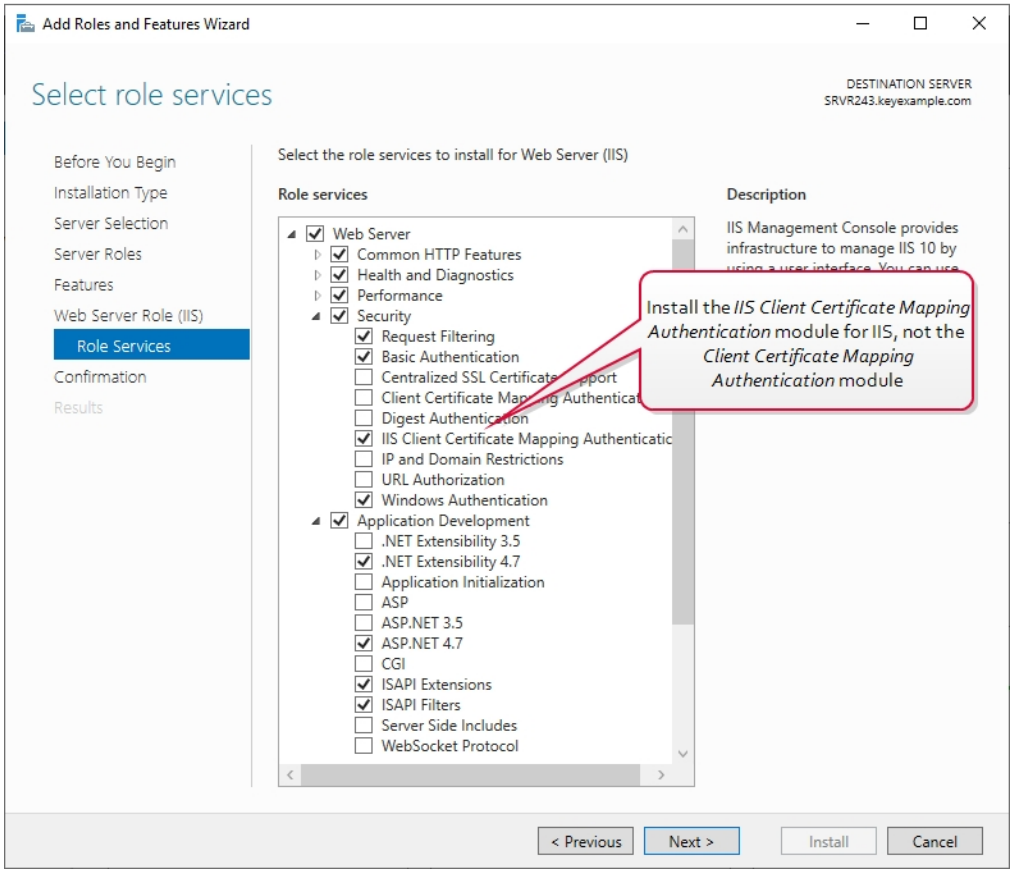


Figure 603: IIS Module for Client Certificate Authentication

The PowerShell command to install the appropriate module is :

```
Add-WindowsFeature Web-Cert-Auth
```

Configure Certificate Authentication and SSL Settings in IIS

Make the following changes in the IIS Management console on the Keyfactor Command server:

1. In the IIS Management console, highlight the server name on the left and open Authentication. Make sure *Anonymous Authentication* is the only enabled method.

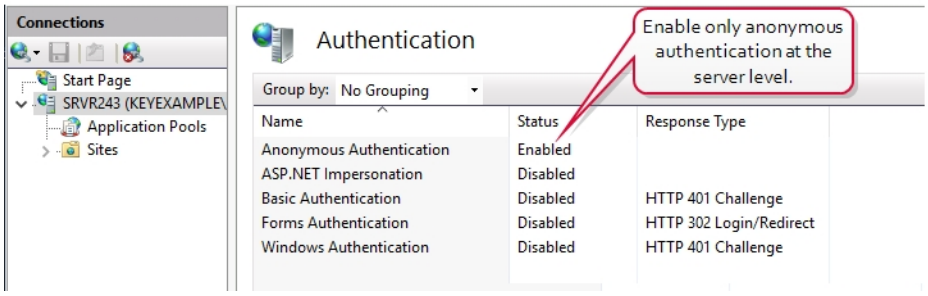


Figure 604: Configure only Anonymous Authentication at the Server Level in IIS

2. In the IIS Management console, drill down into sites and into the Default Web Site (or other web site if your Keyfactor Command instance has been installed in an alternate web site). Under the Default Web Site, locate the KeyfactorAgents application and open Authentication for this. Disable all the authentication methods shown here.

Figure 605: Disable Authentication Methods at the Application Level in IIS

Tip: If your KeyfactorAgents endpoint is running on a standalone server with no other Keyfactor roles, you should also disable all authentication methods at the Default Web Site level as in step two. If your server holds other Keyfactor roles, leave this in the default configuration with Anonymous being the only authentication method enabled as in step one.

3. In the IIS Management console, open SSL Settings for the KeyfactorAgents application. Check the **Require SSL** box and select either **Require** or **Accept** for *Client certificates*.

Important: Only selected **Require** if your are only using orchestrators that support client certificate authentication and plan to configure all of them for certificate authentication.

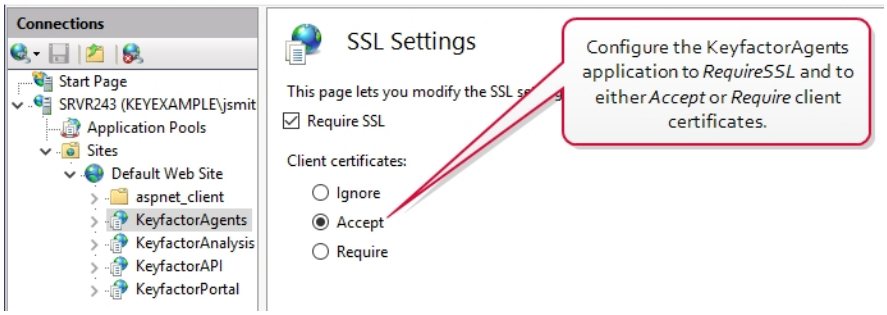


Figure 606: Configure SSL Settings in IIS for Client Certificate Authentication

Tip: If your KeyfactorAgents endpoint is running on a standalone server with no other Keyfactor roles, you may also configure your server to **Require** or **Accept** for *Client certificates* at the Default Web Site level. It is good security practice to check the *Require SSL* box. If your KeyfactorAgents endpoint is running on a server with other Keyfactor roles, you do not need to accept client certificates at this level and should not require them at this level.

Configure Basic Authentication Credentials in IIS

Make the following changes in the IIS Management console on the Keyfactor Command server:

1. In the IIS Management console, drill down to the Default Web Site (or other web site if your Keyfactor Command instance has been installed in an alternate web site). In the Default Web Site, open the Configuration Editor tool.

2. In the Configuration Editor tool at the Default Web Site level, browse to:

   system.webServer/security/authentication/iisClientCertificateMappingAuthentication

   Important: Don't be tempted to configure this setting only at the application level (KeyfactorAgents) rather than at the Default Web Site level. It will only work if configured at the Default Web Site level and then enabled at the application level.

3. In the configurations for IIS Client Certificate Mapping Authentication, set the *defaultLogonDomain* to your forest root. Set the *manyToOneCertificateMappingsEnabled* option to *True* and the *oneToOneCertificateMappingEnabled* option to *False*. Click the dots to the right of the *manyToOneMappings* setting to open details for this setting.

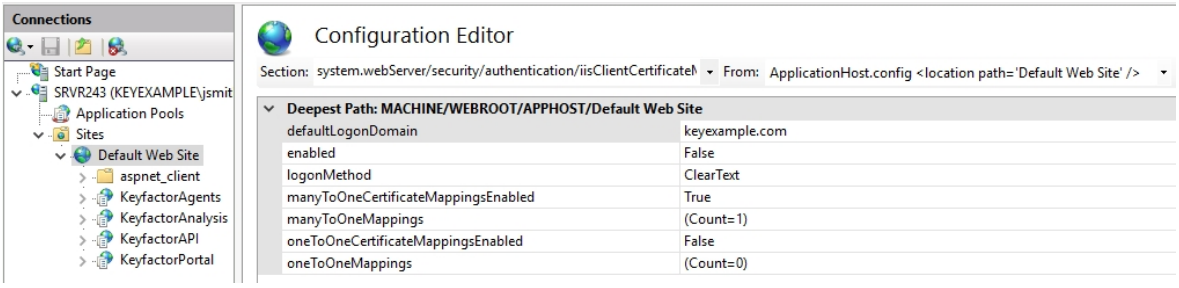


Figure 607: Configure IIS Client Certificate Mapping Authentication for the Default Web Site

4. In the Collection Editor for the manytoOneMappings, click **Add** and enter appropriate values for the properties. The service account entered here will be used as the identity in Keyfactor Command of all orchestrators that authenticate via client certificate.

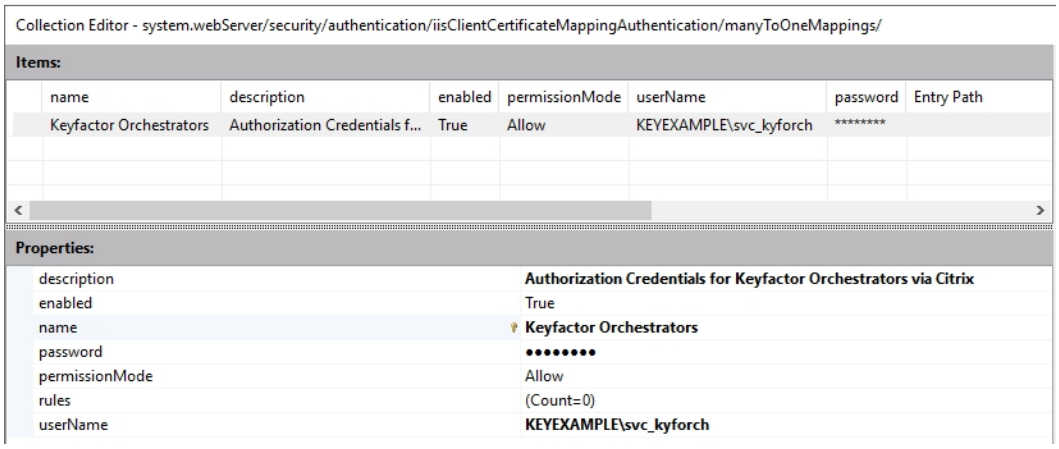


Figure 608: Configure Authorization Credentials for Keyfactor Orchestrators

5. In the IIS Management console,drill down into sites and into the Default Web Site (or other web site if your Keyfactor Command instance has been installed in an alternate web site). Under the Default Web Site, locate the KeyfactorAgents application and open the Configuration Editor tool for it.

6. In the Configuration Editor tool at the KeyfactorAgents application level, browse to:

   system.webServer/security/authentication/iisClientCertificateMappingAuthentication

   Enable the mapping authentication option at this level. The configuration should have replicated down from the Default Web Site level.

Configure the Keyfactor Command Application Setting to Use the Certificate from the Header

When the orchestrator is configured to use a client certificate to authenticate to a proxy and then the proxy is configured to use a separate client certificate to authenticate to the Keyfactor Command server, authentication to the Keyfactor Command application should be done using the original certificate from the orchestrator, not the certificate inserted in the process at the proxy level. This is done by including the original certificate from the orchestrator in the request header to Keyfactor Command. To assure that Keyfactor Command gives priority to this certificate and not the certificate the proxy uses to authenticate, set the Keyfactor Command authentication application setting *Always Use Certificate from Header* to *True*.

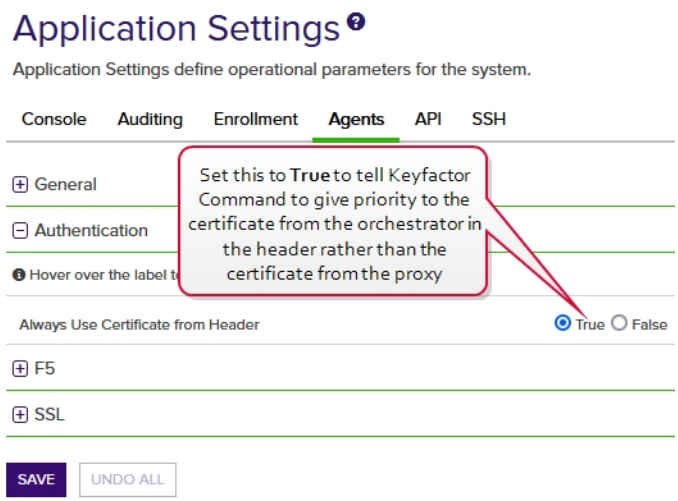


Figure 609: Configure Application Setting in Keyfactor Command to use the Header Certificate

Tip: In some proxy configurations, the proxy may be unable to negotiate the client certificate handshake with IIS. IIS won't ask directly for a client certificate, and if, during the handshake, the proxy doesn't send one, the client authentication will fail. If this occurs, you may need to enable client certificate negotiation at a lower level below IIS. To do this:

1. On the Keyfactor Command server, open a command prompt using the "Run as administrator" option.

2. Execute the following command to output the current configuration for SSL certificate bindings:

   ```
   netsh http show sslcert
   ```

   Output from this command will look something like this (you may see multiple sections if you have multiple web sites on the server):

```
SSL Certificate bindings:
-------------------------
IP:port                       : 0.0.0.0:443
Certificate Hash              : 649dfa6df693583f609af499fe4237f2c1d64224
Application ID                : {4dc3e181-e14b-4a21-b022-59fc669b0914}
Certificate Store Name        : My
Verify Client Certificate Revocation : Enabled
Verify Revocation Using Cached Client Certificate Only : Disabled
Usage Check                   : Enabled
Revocation Freshness Time     : 0
URL Retrieval Timeout         : 0
Ctl Identifier                : (null)
Ctl Store Name                : (null)
DS Mapper Usage               : Enabled
Negotiate Client Certificate : Disabled
Reject Connections            : Disabled
Disable HTTP2                 : Not Set
Disable QUIC                  : Not Set
Disable TLS1.2                : Not Set
Disable TLS1.3                : Not Set
Disable OCSP Stapling         : Not Set
Disable Legacy TLS Versions   : Not Set
```

3. Look at the value for the *Negotiate Client Certificate* setting for the web site on which Keyfactor Command is installed. If the value is *Disabled*, retrieve from the output the values for the *IP:port*, *Certificate Hash*, and *Application ID*.

4. Execute the following commands to remove and re-add the *IP:port* with *Negotiate Client Certificate* enabled (referencing the correct values for ipport, certhash, and appid):

    ```
    netsh http delete sslcert ipport=0.0.0.0:443
    netsh http add sslcert ipport=0.0.0.0:443
    certhash=649dfa6df693583f609af499fe4237f2c1d64224 appid={4dc3e181-e14b-4a21-
    b022-59fc669b0914} clientcertnegotiation=enable
    ```

5. Execute the *show* command again to confirm that the setting is now shown as enabled.

6. Restart the IIS services (iisreset) and try the certificate authentication again.

5.6.3 Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication with Certificates Stored in Active Directory

The Keyfactor Universal Orchestrator can be configured to support client certificate authentication by acquiring a certificate for the Keyfactor Command connect service account user or machine

account of the orchestrator and storing it in Active Directory and then providing the associated Active Directory credentials to authenticate to Keyfactor Command. This has an advantage over the reverse proxy method (see Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC on page 3023) in that a username and password do not need to be stored anywhere (other than in Active Directory). This method does have a heavier reliance on Active Directory.

Complete the following steps and then configure the orchestrator to enable client certificate authentication as per the installation instructions (see -ClientCertificate (Client Certificate Authentication) on page 2904 or Install the Universal Orchestrator on a Linux Server on page 2912).

Tip: Using this method, you do not necessarily need to configure certificate authentication in Keyfactor Command, unlike for the proxy method (see Appendix - Set up the Universal Orchestrator to Use Client Certificate Authentication via a Reverse Proxy: Citrix ADC on page 3023), since the certificate authentication is occurring at the IIS layer before the request reaches Keyfactor Command. You may wish to configure certificate authentication in Keyfactor Command to allow Keyfactor Command to monitor certificate authentication and to support automated certificate renewal (see Register a Client Certificate Renewal Extension on page 2961). If you enable certificate authentication in Keyfactor Command with this method, you will need to provide a value in the *Certificate Authentication HTTP Header* field. This header field is used to pass the certificate contents to Keyfactor Command command in instances when the certificate is not used directly (such as in the reverse proxy scenario). The value is required when configuring certificate authentication in Keyfactor Command, but since for this method you do not need to extract the certificate from the header, the value you set here is unimportant.

Important: If you do opt to enable certificate authentication in Keyfactor Command, be aware that this will force all orchestrators to use certificate authentication when communicating with Keyfactor Command on the configured server.

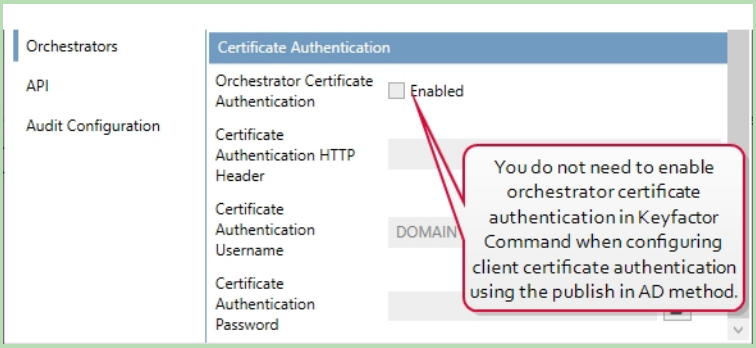


Figure 610: Client Certificate Authentication with AD Storage Does Not Require Certificate Authentication Configuration in Keyfactor Command

Note: The following instructions assume that your Keyfactor Command server is already installed and configured with an SSL certificate that is trusted in your environment. If this is not the case, this will also need to be done.

Install the Required Windows Module

On your Keyfactor Command server, install the following additional module:

- *Client Certificate Mapping Authentication* (rather than *IIS Client Certificate Mapping Authentication*)

If you have more than one Keyfactor Command server with separated roles, this only needs to be installed on the server accepting traffic to the /KeyfactorAgents web application.

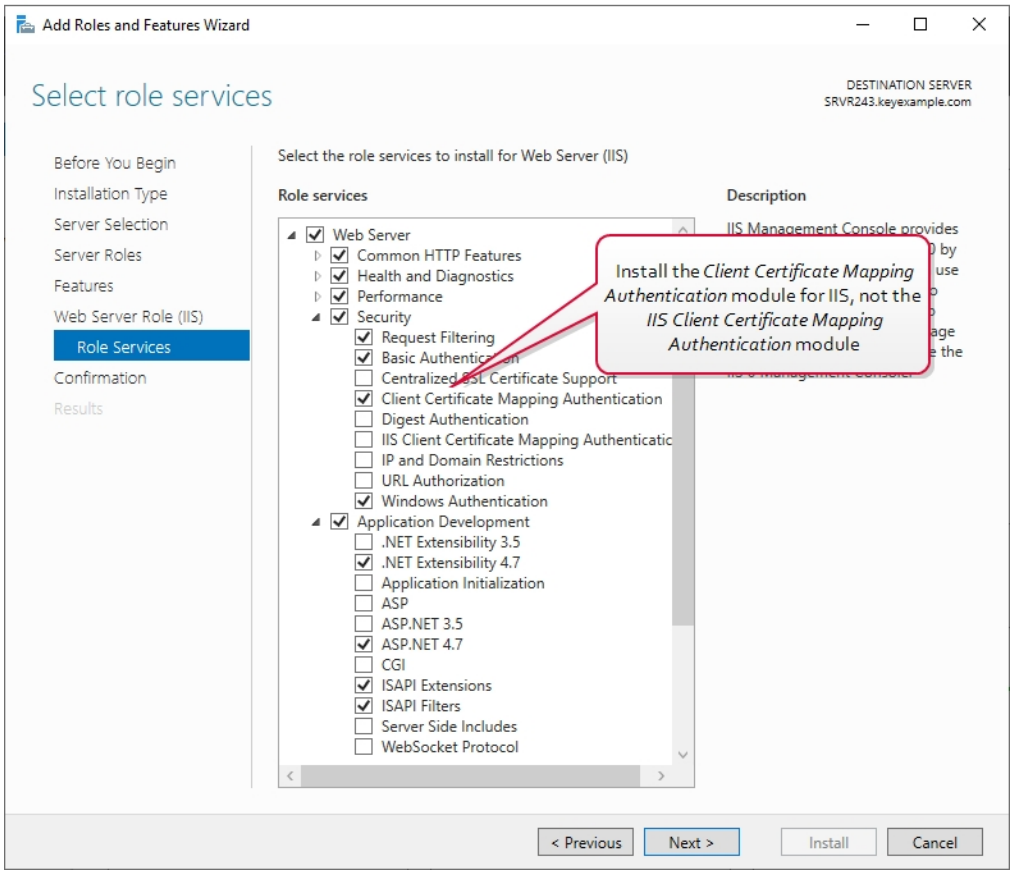


Figure 611: IIS Module for Client Certificate Authentication with AD Storage

The PowerShell command to install the appropriate module is :

```
Add-WindowsFeature Web-Client-Auth
```

Configure Certificate Authentication and SSL Settings in IIS

Make the following changes in the IIS Management console on the Keyfactor Command server:

1. In the IIS Management console, highlight the server name on the left and open Authentication. Change the status of *Active Directory Client Certificate Authentication* to **Enabled**.

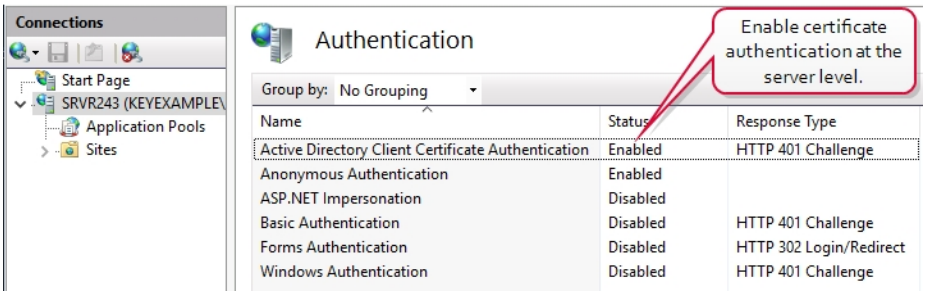


Figure 612: Configure Client Certificate Authentication at the Server Level in IIS

2. In the IIS Management console, drill down into sites and into the Default Web Site (or other web site if your Keyfactor Command instance has been installed in an alternate web site). Under the Default Web Site, locate the KeyfactorAgents application and open Authentication for this. Disable all the authentication methods shown here. The *Active Directory Client Certificate Authentication* method does not appear here.

Figure 613: Disable Authentication Methods at the Application Level in IIS

Tip: At the Default Web Site level, the only authentication method that should be enabled is Anonymous. This should not be changed.

3. In the IIS Management console, open SSL Settings for the KeyfactorAgents application. Check the **Require SSL** box and select either **Require** or **Accept** for *Client certificates*.

Important: Only selected **Require** if your are only using orchestrators that support client certificate authentication and plan to configure all of them for certificate authentication.

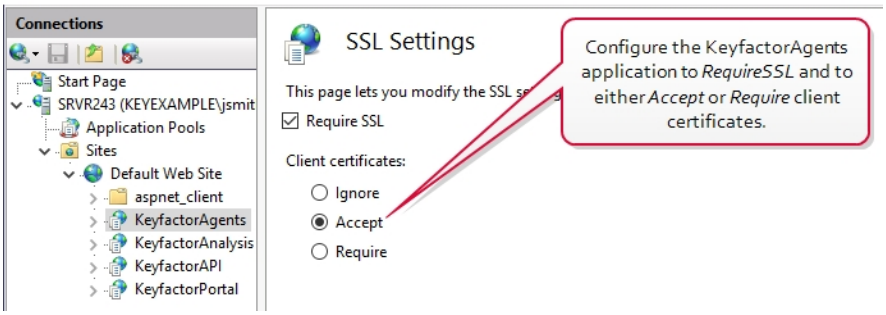


Figure 614: Configure SSL Settings in IIS for Client Certificate Authentication

Tip: At the Default Web Site level, it is good security practice to check the *Require SSL* box, but you do not need to accept client certificates at this level and should not require them at this level.

Create a Certificate Template for Orchestrator Certificates

This method of certificate authentication functions by sending a client certificate from the orchestrator to IIS on the Keyfactor Command server, where IIS does a lookup in Active Directory to determine what Active Directory user is associated with that certificate and then turns around and uses that identity to connect to Keyfactor Command. In order for the certificate to the associated with the Active Directory identity, it must be enrolled using a template that has the *Publish certificate in Active Directory* option enabled.

To create the certificate template that will be used for orchestrator client authentication certificates, start by duplicating a template with a *Computer* subject type. In addition to any standards for your environment, the templates needs:

- The *Publish certificate in Active Directory* box checked.

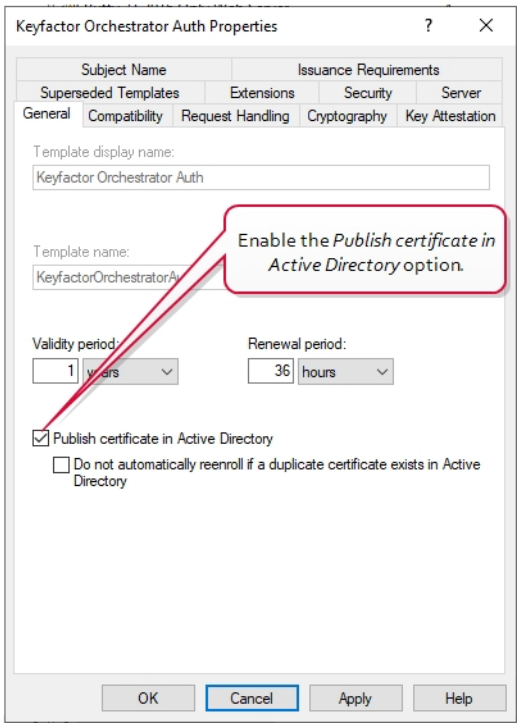


Figure 615: Microsoft Certificate Template General for Client Authentication Certificate

- A key usage that includes Digital Signature.

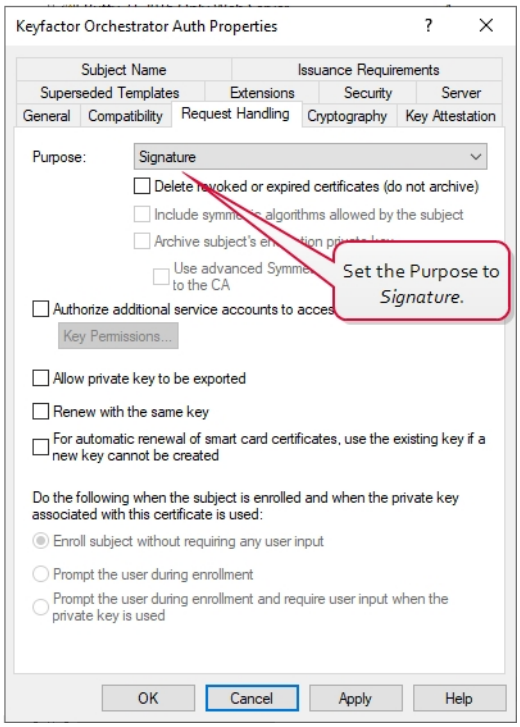


Figure 616: Microsoft Certificate Template Request Handling for Client Authentication Certificate

- An extended key usage (EKU) of Client Authentication.

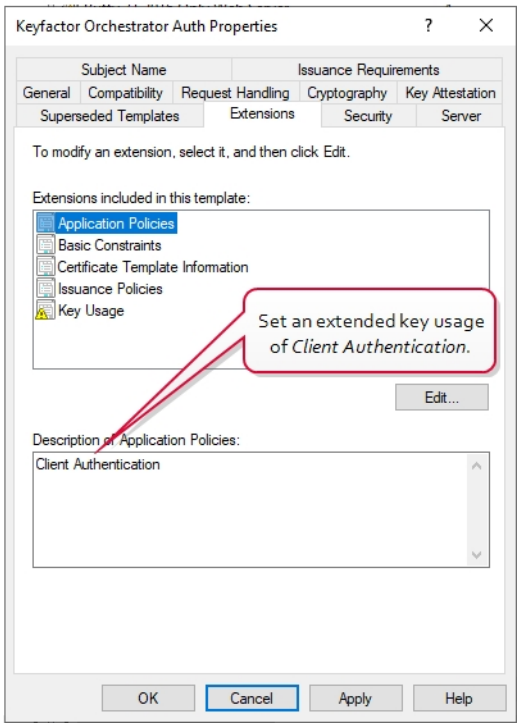


Figure 617: Microsoft Certificate Template Application Policies for Client Authentication Certificate

- Enroll permissions for either the service account that the orchestrator will run as or the machine account for the orchestrator machine (see).

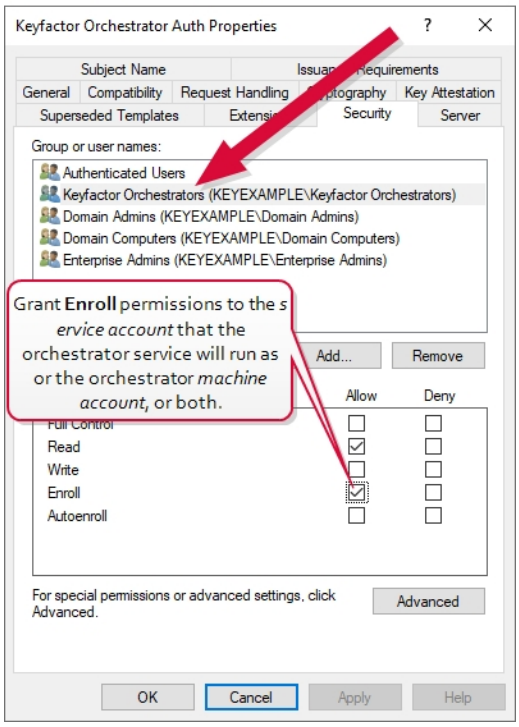


Figure 618: Microsoft Certificate Template Security for Client Authentication Certificate

Enroll for a Client Authentication Certificate

To acquire a certificate for use by the Universal Orchestrator using a Microsoft CA, first create a template using the appropriate configurations as described above and make it available for enrollment on a CA to which the Universal Orchestrator machine has access. If you plan to enroll for the certificate through Keyfactor Command, you will also need to enable the template for enrollment in Keyfactor Command.

You can enroll for a client authentication certificate for the orchestrator in a variety of ways. The certificate needs to be installed in the **local computer** personal store on the Windows server on which the orchestrator is installed. Some possible ways to do this are:

- Use Keyfactor Command to enroll for the certificate using the PFX enrollment method and then import the PFX file on the orchestrator server. If you select this method, you will need to login to the Keyfactor Command Management Portal as the orchestrator service account being used on the Keyfactor Command side of the fence (see Create Service Accounts for the Universal Orchestrator on page 2884) in order to enroll for the certificate in the correct context or use the Keyfactor API to submit a request in a specific user context (see PFX Enrollment in Keyfactor Command Using a PowerShell Script on the next page). The orchestrator service account will need enroll permissions on the CA, on the template, and in Keyfactor Command.
- Use IIS or the certificates MMC on the orchestrator server to generate a CSR, use the Keyfactor Command CSR enrollment method to enroll for a certificate using the CSR, and then import the CSR on the orchestrator server, marrying it with the private key generated on the

server. If you select this method, you will need to login to the Keyfactor Command Management Portal as the orchestrator service account being used on the Keyfactor Command side of the fence (see Create Service Accounts for the Universal Orchestrator on page 2884) in order to enroll for the certificate in the correct context. The orchestrator service account will need enroll permissions on the CA, on the template, and in Keyfactor Command.

- If there is an existing Universal Orchestrator on the server already running and communicating with Keyfactor Command, use the Keyfactor Command PFX enrollment method and push the certificate out to the certificate store on the orchestrator server using Keyfactor Command. If you select this method, you will need to login to the Keyfactor Command Management Portal as the orchestrator service account being used on the Keyfactor Command side of the fence (see Create Service Accounts for the Universal Orchestrator on page 2884) in order to enroll for the certificate in the correct context or use the Keyfactor API to submit a request in a specific user context (see PFX Enrollment in Keyfactor Command Using a PowerShell Script below). The orchestrator service account will need enroll permissions on the CA, on the template, and in Keyfactor Command.
- Use the Microsoft MMC on the orchestrator server to enroll for a certificate. If you select this method, the orchestrator will connect to Keyfactor Command using the orchestrator machine account rather than an Active Directory user account. The orchestrator machine account will need enroll permissions on the CA and on the template. This method will only work for servers joined to the same Active Directory forest in which Keyfactor Command is installed.

PFX Enrollment in Keyfactor Command Using a PowerShell Script

To enroll for a certificate using the PFX enrollment method in Keyfactor Command, you can either do this in the Keyfactor Command Management Portal while logged in as the orchestrator service account or with a PowerShell script. In either case, the orchestrator service account will need PFX enroll permissions in Keyfactor Command. Below is a sample PowerShell script. Once the PFX file has been generated, import it into the local machine store on the orchestrator server.

Tip: The service account you provide in the PowerShell script is the service account used to provide a connection from the orchestrator to Keyfactor Command. This is not necessarily the same service account that runs the orchestrator service on the orchestrator server. For an orchestrator in a separate forest from Keyfactor Command, this would be a service account in the Keyfactor Command forest, not the orchestrator forest. See Create Service Accounts for the Universal Orchestrator on page 2884.

```
#Set variables with the username and password for the orchestrator service account
$orchUsername = 'KEYEXAMPLE\svc_kyforch'
$orchPassword = 'MySecureServiceAccountPassword'
$pair = "$($orchUsername):$($orchPassword)"

# Base-64 encode the service account credentials
$encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
```

```
$UTCTime = (Get-Date).ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ssZ")
$keyfactorServer = 'keyfactor.keyexample.com' # FQDN of the Keyfactor Command server
$caName = 'corpca01.keyexample.com\CorpIssuing01' # CA to use for the enrollment
$templateName = 'KeyfactorOrchestratorAuth' # Template to use for the enrollment
$certSubject = 'Orchestrator Cert Auth' # Using a template that is configured to build from AD will
cause this subject to be replaced
$pfxPassword = 'MySecurePFXPassword' # Password for the resulting PFX file
$outputFile = 'C:\stuff\OrchCertAuth.pfx' # Path and file name for the PFX file to be generated

$basicAuthValue = "Basic $encodedCreds"

$headers = @{
"Authorization"=$basicAuthValue
"Accept"="application/json"
"x-keyfactor-requested-with"="APIClient"
"x-certificateformat"="PFX"
}

$body = @{
"Password" = "$pfxPassword"
"Subject" = "$certSubject"
"IncludeChain" = "true"
"CertificateAuthority" = "$caName"
"Timestamp" = "$UTCTime"
"Template" = "$templateName"
}

# Output response as a PFX file
$response = Invoke-WebRequest -Uri "https://$keyfactorServer/KeyfactorAPI/Enrollment/PFX" -Meth-
od:Post -Headers $headers -ContentType "application/json" -Body ($body|ConvertTo-Json) -ErrorAc-
tion:Stop -TimeoutSec 60
$ResponseContent = $response.Content | ConvertFrom-Json
$bytes = [Convert]::FromBase64String($ResponseContent.CertificateInformation.Pkcs12Blob)
[IO.File]::WriteAllBytes($outputFile, $bytes)
```

PFX Enrollment and Deployment in Keyfactor Command Using a PowerShell Script

To enroll for a certificate using the PFX enrollment method in Keyfactor Command and deploy it to the orchestrator server using Keyfactor Command, you can either do this in the Keyfactor Command Management Portal while logged in as the orchestrator service account or with a PowerShell script. In either case, the orchestrator service account will need PFX enroll permissions and certificate store management permissions in Keyfactor Command. Below is a sample PowerShell script. This solution is only an option if your orchestrator is already up and running and successfully authenticating to Keyfactor Command using standard authentication (or previously configured

certificate authentication).

Tip: The service account you provide in the PowerShell script is the service account used to provide a connection from the orchestrator to Keyfactor Command. This is not necessarily the same service account that runs the orchestrator service on the orchestrator server. For an orchestrator in a separate forest from Keyfactor Command, this would be a service account in the Keyfactor Command forest, not the orchestrator forest. See Create Service Accounts for the Universal Orchestrator on page 2884.

```
#Set variables with the username and password for the orchestrator service account
$orchUsername = 'KEYEXAMPLE\svc_kyforch'
$orchPassword = 'MySecureServiceAccountPassword'
$pair = "$($orchUsername):$($orchPassword)"

# Base-64 encode the service account credentials
$encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))

$UTCTime = (Get-Date).ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ssZ")
$keyfactorServer = 'keyfactor.keyexample.com' # FQDN of the Keyfactor Command server
$storeName = 'websrvr38.keyexample.com' # FQDN of the orchestrator server as defined as a certificate store in Keyfactor Command
$caName = 'corpca01.keyexample.com\CorpIssuing01' # CA to use for the enrollment
$templateName = 'KeyfactorOrchestratorAuth' # Template to use for the enrollment
$certSubject = 'Orchestrator Cert Auth' # Using a template that is configured to build from AD will cause this subject to be replaced

$basicAuthValue = "Basic $encodedCreds"

$enrollHeaders = @{
   "Authorization"=$basicAuthValue
   "Accept"="application/json"
   "x-keyfactor-requested-with"="APIClient
   "x-certificateformat"="Store"
}

$deployHeaders = @{
   "Authorization"=$basicAuthValue
   "Accept"="application/json"
   "x-keyfactor-requested-with"="APIClient"
}

$enrollBody = @{
```

```
   "Subject" = "$certSubject"
   "IncludeChain" = "true"
   "CertificateAuthority" = "$caName"
   "Timestamp" = "$UTCTime"
   "Template" = "$templateName"
}

# Enroll for a certificate using the PFX enrollment method and retrieve the certificate ID from the
response (as part of the content)
$enrollResponse = Invoke-WebRequest -Uri "https://$keyfactorServer/KeyfactorAPI/Enrollment/PFX" -
Method:Post -Headers $enrollHeaders -ContentType "application/json" -Body ($enrollBody|ConvertTo-
Json) -ErrorAction:Stop -TimeoutSec 60
$enrollContent = $enrollResponse.Content | ConvertFrom-Json

# Get the store GUID for the certificate store specified by the client machine name in the query
string with the storeName variable
$storeInfo = Invoke-WebRequest -Uri "https://$key-
factorServer/KeyfactorAPI/CertificateStores?certificateStoreQuery.queryString=ClientMachine%20-
eq%20%22$storeName%22" -Method:Get -Headers $deployHeaders -ContentType "application/json" -
ErrorAction:Stop -TimeoutSec 60
$storeContent = $storeInfo.Content | ConvertFrom-Json
$storeGUID = $storeContent.Id

$deployBody = @{
   "StoreIds" = @( "$storeGUID" )
   "StoreTypes" = @(
      @{
         "StoreTypeId" = 6 # Store type 6 is IIS personal
         "Overwrite" = "false"
      }
   )
   "CertificateId" = $enrollContent.CertificateInformation.KeyfactorId
}

# Deploy certificate to certificate store
Invoke-WebRequest -Uri "https://$keyfactorServer/KeyfactorAPI/Enrollment/PFX/Deploy" -Method:Post -
Headers $deployHeaders -ContentType "application/json" -Body ($deployBody|ConvertTo-Json) -ErrorAc-
tion:Stop -TimeoutSec 60
```

MMC Enrollment

To enroll for a certificate using the MMC:

1. On the Universal Orchestrator machine, do one of following:
    - Using the GUI:
        a. Open an empty instance of the Microsoft Management Console (MMC).
        b. Choose **File->Add/Remove Snap-in...**.
        c. In the *Available snap-ins* column, highlight **Certificates** and click **Add**.
        d. In the Certificates snap-in popup, choose the radio button for Computer account, click **Next**, accept the default of Local computer, and click **Finish**.
        e. Click **OK** to close the Add or Remove Snap-ins dialog.
    - Using the command line:
        a. Open a command prompt using the "Run as administrator" option.
        b. Within the command prompt type the following to open the certificates MMC:

            ```
            certlm.msc
            ```

2. Drill down to the Personal folder under **Certificates** for the Local Computer, right-click, and choose **All Tasks->Request New Certificate...**.
3. Follow the certificate enrollment wizard, selecting the template you created for orchestrator certificate authentication and providing any required information.

Grant the Service Account Certificate Private Key Permissions

Whichever method you decide to use to acquire the client authentication certificate for the orchestrator, you will need to grant the Universal Orchestrator service account—the account that the orchestrator service is running as on the server—permissions to read the private key of that certificate.

> **Tip:** If the service account is a member of the local administrators group, this step may not be necessary, since the local administrators group is typically granted these permissions automatically.

To grant private key permissions on the certificate using the MMC:

1. On the Universal Orchestrator machine, do one of following:
    - Using the GUI:
        a. Open an empty instance of the Microsoft Management Console (MMC).
        b. Choose **File->Add/Remove Snap-in...**.
        c. In the *Available snap-ins* column, highlight **Certificates** and click **Add**.
        d. In the Certificates snap-in popup, choose the radio button for Computer account,

click **Next**, accept the default of Local computer, and click **Finish**.

e. Click **OK** to close the Add or Remove Snap-ins dialog.

- Using the command line:
  a. Open a command prompt using the "Run as administrator" option.
  b. Within the command prompt type the following to open the certificates MMC:

     ```
     certlm.msc
     ```

2. Drill down to the Personal folder under **Certificates** for the Local Computer to locate the certificate.
3. Highlight the certificate and choose **All Tasks->Manage Private Keys...**.
4. In the Permissions for private keys dialog, click **Add**, add the service account under which the Universal Orchestrator is running (created as per Create Service Accounts for the Universal Orchestrator on page 2884), and grant that service account **Read** but not **Full control** permissions. Click **OK** to save.

Tip: If you receive the following error when selecting your certificate in the orchestrator configuration wizard:

```
The request was aborted: Could not create SSL/TLS secure channel.
```

- Confirm that the orchestrator server trusts the root and issuing certificates for the SSL certificate on the Keyfactor Command server and the client authentication certificate you are trying to use (see Configure Certificate Root Trust for the Universal Orchestrator on page 2888).
- Confirm that the orchestrator server has access to the CRLs for both the SSL certificate on the Keyfactor Command server and the client authentication certificate you are trying to use and that these CRLs are valid.
- Confirm that you have granted the service account under which the orchestrator service runs private key permissions on the client authentication certificate.

5.6.4 Appendix - Set up the Universal Orchestrator to Use a Forwarding Proxy

Typically with services that use a forwarding proxy, there is a specific proxy configuration done within the application, but the Universal Orchestrator doesn't have such a configuration. Instead, it makes use of an environment variable to retrieve this information on either Windows or Linux.

On Windows, configure a system environment variable of either HTTP_PROXY or HTTPS_PROXY (this is not case sensitive on Windows) pointing to your proxy's URL, including port, then restart the Universal Orchestrator service if the orchestrator is already installed.

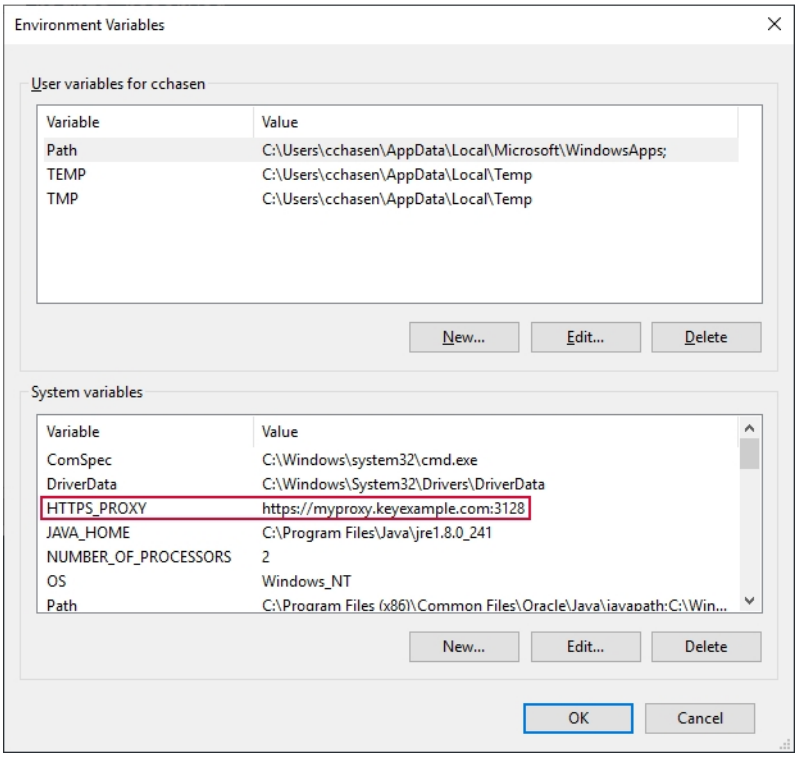


Figure 619: System Environment Variable to Define a Proxy URL for Use by the Universal Orchestrator on Windows

On Linux, there are multiple approaches to setting an environment variable. One method for setting a system-wide environment variable that will be retained after reboot is to add an environment variable statement to the /etc/environment file using a command similar to the following (as root):

```
echo https_proxy=https://myproxy.keyexample.com:3128/" >> /etc/environment
```

After setting the environment variable, restart the Universal Orchestrator service if the orchestrator has already been installed.

Note: If you've configured an HTTPS_PROXY environment variable because you're using a secure channel to communicate with Keyfactor Command (SSL), you will most likely also need an HTTP_PROXY environment variable for the orchestrator to do revocation status (CRL) checking unless you disable revocation status checking.

6.0 Release Notes & Upgrading

The Keyfactor Command suite of documentation is released as both major releases, with version numbers ending in zero, and minor releases, with incremental fixes and updates following the major release. When reviewing release notes, be sure to review those for both the minor releases and their corresponding major release.

Upgrade instructions are included for Keyfactor-hosted and self-hosted installation (see Upgrade Overviews below).

6.1 Upgrade Overviews

The *Keyfactor Command Upgrade Overview* is provided in two formats for different users. Follow the appropriate upgrade instructions for your configuration.

- Upgrade Overview - Keyfactor-Hosted below

  Use this guide if you do any of the following:

  - Consume Keyfactor Command certificate lifecycle automation as a service hosted by Keyfactor
  - Consume a managed PKI hosted by Keyfactor

- Upgrade Overview - Self-Hosted on page 3055

  Use this guide if you have any of the following:

  - Have deployed Keyfactor Command on premise in your data center or cloud
  - Have deployed the Keyfactor CA Policy Module and associated CA policy handlers on premise in your data center or cloud

6.1.1 Upgrade Overview - Keyfactor-Hosted

The Keyfactor Command solution by Keyfactor allows organizations to issue and manage certificates across enterprise infrastructures. For a comprehensive description of the components that make up Keyfactor Command, see Logical Architecture on page 2696 and Installing Orchestrators on page 2875. There are also Keyfactor installation guides for third-party CA gateways that interface with Keyfactor Command. For an overview of the key new features in the latest version of Keyfactor Command, see the Release Notes & Upgrading above.

This document provides guidance to help you prepare for and complete an upgrade. The Keyfactor Command server software will be upgraded for you. If you need assistance upgrading other components, please contact your Client Success Manager.

Before upgrading to a major version, Keyfactor recommends first upgrading to the final incremental version of the previous major version—completing both the software installation and configuration with the configuration wizard—for the optimal upgrade experience. For example, if you are currently on version 9.4 and want to upgrade to version 10.0 or later, you should upgrade first to version

9.10.1 (the final incremental version of 9.x) before upgrading. Contact your Client Success Manager for more information.

6.1.1.1 Upgrading

Most Keyfactor Command upgrades are brief with a minimum of changes to existing user accounts, groups, CA templates, firewall settings, etc. The prerequisites have not materially changed from previous versions and the current version can generally be installed using the same hardware and existing instances of the supporting software. The upgrade process is often completed within three to four hours, including the time spent by your Keyfactor representative to upgrade your hosted environment.

The overall task flow consists of the following steps:

Upgrade of the Server Software

The Keyfactor Command server software will be installed and configured for you. Once this is complete, you may upgrade any orchestrators and gateways in your environment.

Upgrade the Keyfactor Universal Orchestrator

In many cases the Keyfactor Universal Orchestrator can be installed over the existing installation without uninstalling the previous version. For specific upgrade guidance, see Upgrading the Universal Orchestrator on page 2895.

Update from Windows Orchestrators

Support for the Keyfactor Windows Orchestrator was deprecated in Keyfactor Command release 11.0. All uses of the Keyfactor Windows Orchestrator should be updated to the Keyfactor Universal Orchestrator. The Keyfactor Universal Orchestrator replaces the Keyfactor Windows Orchestrator and runs on both Windows or Linux servers. The following functions that were part of the Keyfactor Windows Orchestrator are supported in the Keyfactor Universal Orchestrator with custom extensions:

- Interact with F5 devices for certificate management
- Interact with NetScaler devices for certificate management
- Interact with Amazon Web Services (AWS) resources for certificate management
- Interact with Windows certificate stores and IIS

For more information about using custom extensions with the Keyfactor Universal Orchestrator, see Installing Custom-Built Extensions on page 2940.

> **Important:** The Keyfactor Universal Orchestrator is only compatible with Keyfactor Command version 9.0 or later. The current version of the Keyfactor Universal Orchestrator is 11.1 and requires .NET 6.

If you're upgrading from a version of Keyfactor Command prior to 8.0, you will need to update any Windows Orchestrators (a.k.a. Windows Agents) that are used for SSL scanning to support the current scanning architecture. Install and configure the Keyfactor Universal Orchestrator software (see Upgrading the Universal Orchestrator on page 2895).

Note: The orchestrator endpoint location changed for Keyfactor Command release 6 and may need to be modified in your orchestrator endpoint configuration—from CMSAgents to KeyfactorAgents.

Cloud Gateway

The latest version of the Keyfactor Cloud Gateway—used to support management of certificates in the hosted Keyfactor Command environment—is 23.3 released in mid 2023. If you are already using this version, no configuration changes need to be made. Restart the gateway service to refresh the connection to the upgraded Keyfactor Command instance.

If you're using a recent version of the gateway (20.6 or newer), you don't need to upgrade the gateway unless the gateway contains a change that's needed in your environment. See the gateway release notes in the *Keyfactor Cloud Gateway Installation & Configuration Guide* to review some of the recent changes.

In most cases, the Keyfactor gateway software can be installed over the existing software installation without uninstalling the previous version. Review the configuration for your gateway, and then install and configure the software as per the *Keyfactor Cloud Gateway Installation & Configuration Guide*, retaining the same installation location.

EJBCA CA Gateway

If you're using an EJBCA gateway and wish to make use of the new feature in Keyfactor Command for native support of EJBCA CAs, you will need to follow the EJBCA gateway upgrade process to unlink the EJBCA certificates in your Keyfactor Command database from your EJBCA gateway CA to enable them to be relinked to a native CA configured in Keyfactor Command. For more information, contact Keyfactor support.

Other CA Gateways

In most cases, the Keyfactor gateway software can be installed over the existing software installation without uninstalling the previous version. Review the configuration for your gateway, and then install and configure the software as per the Keyfactor gateway guide for the particular gateway, retaining the same installation location. The gateway configuration wizard has significantly changed in recent releases for many of the gateways, which may require modification to your configuration.

Tip: New versions of CA gateways are not necessarily released at the same time as new versions of Keyfactor Command and so gateways may not need upgrading at the same time as Keyfactor Command.

API

Important: The Classic API, also known as the CMS API or CMSAPI, has been deprecated in Keyfactor Command version 11.0. Customers should migrate all uses of the Classic API to the Keyfactor API prior to upgrading to Keyfactor Command version 11.

Please see the latest Release Notes & Upgrading on page 3051 if you are using any custom scripts that leverage the Keyfactor API.

Post-Install Configuration and Testing

See Post-Upgrade Steps below.

The bulk of the time upgrading will be spent verifying that all functions and configurations have correctly carried over and the upgraded instance is performing correctly.

6.1.1.2 Post-Upgrade Steps

There is no particular order in which the tasks on the following pages must be accomplished.

Tip: If, following the upgrade, you open a page in the Keyfactor Command Management Portal and find it unexpectedly blank or otherwise displaying incorrectly, try refreshing the page with a CTRL-F5. If this doesn't resolve the problem, try clearing the browser cache and then reloading the page. It may be helpful to advise all end users to do this following an upgrade.

Testing

Once everything is up and running again, confirm that the following features are operating correctly:

- Does the Keyfactor Command Management Portal load correctly?
- Run a report in the Keyfactor Command Management Portal to confirm that the connectivity to LogiAnalytics is operating correctly.
- Issue a certificate in the Keyfactor Command Management Portal to confirm connectivity to CAs.

Post-Install Configuration

If you are upgrading from any release of Keyfactor Command version 6 or greater, you may want to make some additional configuration changes post-installation:

- Upgrade any Keyfactor CA gateways in your environment that are based on the AnyGateway. The AnyGateway must be upgraded to at least 22.1 to be compatible with Keyfactor Command 10.0 and later.
- Consider whether you wish to implement Keyfactor Command workflows and whether a Keyfactor Command-level workflow could replace CA-level manager approval for any templates

that are configured to require CA-level manager approval.

- Review the new enrollment default and policy settings for enrollment. Enrollment defaults and polices can be defined at two levels:
  - System-wide settings apply to all enrollments done through Keyfactor Command unless they are overridden by template-specific settings.
  - Template-level settings allow you to modify any established enrollment defaults or policies on a per-template basis.

  There are several settings available for configuration as part of the template policies:
  - Allow Wildcards
  - Allow Public Key Reuse
  - Enforce RFC 2818 Compliance
  - Supported Key Types

  Enrollment defaults allow you to pre-polulate the subject fields in PFX Enrollment and CSR Generation. Users are allowed to override these at enrollment.
- If you're using certificate metadata or regular expressions, optionally define these for each template. Certificate metadata fields and regular expressions can be defined at two levels:
  - System-wide settings apply to all enrollments done through Keyfactor Command unless they are overridden by template-specific settings.
  - Template-level settings allow you to modify any established certificate metadata or regular expressions on a per-template basis (for instance, for a metadata field, whether the field is required, what default value it should provide, or whether to hide the field during enrollment, regardless of system-wide setting).
- The enrollment configuration will have been carried over in the upgrade, however you may want to confirm the configuration of Certificate Authority and Template enrollment (PFX, CSR, and CSR generation) and make any changes.
- Review any template that is configured to require manager approval at the CA level and confirm that a Keyfactor Command private key retention policy in place.
- Review the new reports in the Keyfactor Command Report Manager and add them to the menu or favorite them, if desired.

6.1.2 Upgrade Overview - Self-Hosted

The Keyfactor Command solution by Keyfactor allows organizations to issue and manage certificates across enterprise infrastructures. For a comprehensive description of the components that make up Keyfactor Command, see Logical Architecture on page 2696 and Installing Orchestrators on page 2875. There are also Keyfactor installation guides for third-party CA gateways that interface with Keyfactor Command. For an overview of the key new features in the latest version of Keyfactor Command, see the Release Notes & Upgrading on page 3051.

This document provides guidance to help you prepare for and complete an upgrade. If you need assistance with the upgrade, please contact your Client Success Manager.

Before upgrading to a major version, Keyfactor recommends first upgrading to the final incremental version of the previous major version—completing both the software installation and configuration with the configuration wizard—for the optimal upgrade experience. For example, if you are currently on version 9.4 and want to upgrade to version 10.0 or later, you should upgrade first to version 9.10.1 (the final incremental version of 9.x) before upgrading. Contact your Client Success Manager for more information.

Important: Keyfactor Command version 10.0 and later require an encrypted connection to the SQL server. Upgrades will fail if the SQL server is not correctly configured to support this. See System Requirements on page 3060.

6.1.2.1 Preparing

This section describes the steps that need to be taken prior to a Keyfactor Command upgrade to complete the prerequisites, create any required supporting components, and gather the necessary information to complete the Keyfactor Command upgrade process.

The following are some key things to be aware of and preparation steps that need to be addressed in order to upgrade to version 11.1:

- If you're migrating from Active Directory to an identity provider other than Active Directory in version 11, be aware that it's important to leave Basic Authentication and Windows Authentication disabled in IIS once you enable the OAuth option in the Keyfactor Command configuration. Attempting to use OAuth with Basic Authentication and/or Windows Authentication enabled can result in unexpected access to Keyfactor Command by users with lingering Active Directory permissions in Keyfactor Command from before the upgrade.
- Select substitutable special text tokens provide the ability for parts of the Keyfactor Command system to look up data in Active Directory to use in alerts and workflows. Lookups for these locate the object in Active Directory identified by the user or computer account that requested the certificate from the CA and substitute the contents of the referenced attribute. For example, an alert might reference *requester:displayname* to look up the requester's display namein Active Directory.

  The substitutable special text tokens that begin *requester:* or *principal:* (but not the standalone *requester* token) are only supported when using Active Directory as an identity provider. They cannot be used with other identity providers since there is no Active Directory to query. Customers wishing to use similar functionality with a different identity provider will need to do API calls to query the identity provider, populate the data into a metadata field, and use the metadata field when populating alert and workflow messages.
- In Keyfactor Command version 11.0, the route information for each virtual application (the host name, web site name, use ssl setting, and virtual directory name) has been moved from being stored in the database to being stored in local files on the Keyfactor Command server (one appsettings.json file for each virtual application). On upgrade, the information is copied from the database to each local file. However, if you encounter any issues with the upgrade process, you may be required to re-enter this data. Be sure to make note of the values before beginning your upgrade.

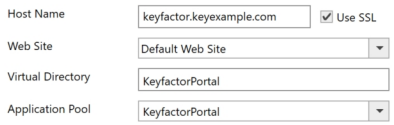


Figure 620: Configuration Wizard Route Information for the Keyfactor Portal

- Keyfactor Command version 10.0 and later by default connects to SQL with an encrypted connection using an SSL certificate configured on your SQL server. Customers should acquire and install an SSL certificate for the SQL server before upgrading to Keyfactor Command version 10.0 or later (see Using SSL to Connect to SQL Server on page 2746). If you would prefer not to use an encrypted channel for your connection to SQL, see Configurable SQL Connection Strings on page 2750.
- Upgrade to SQL Server 2016 CU2 or higher and adjust the database compatibility level if needed. For more information, see System Requirements on page 3060.
- As of Keyfactor Command version 10.0, Windows Server 2016 is no longer supported. The installer will not check your server version nor prevent installation, but the product will not function properly in some instances. Customers should upgrade to Windows Server 2019 or higher before upgrading to Keyfactor Command version 10.0 or later. If you choose to use Server 2016, any PFXs will need to be configured to use SHA1 and 3DES for encryption for use by Keyfactor Command.
- Before upgrading to a major version, Keyfactor recommends first upgrading to the final incremental version of the previous major version—completing both the software installation and configuration with the configuration wizard—for the optimal upgrade experience. For example, if you are currently on version 9.4 and want to upgrade to version 10.0 or later, you should upgrade first to version 9.10.1 (the final incremental version of 9.x) before upgrading. Contact your Client Success Manager for more information.
- If you have any saved certificate collections containing any of the following deprecated certificate search fields, these collections will need to be removed or updated to remove use of these fields that are no longer in version 10.0 and later:
  - *KeyfactorRequestId*
  - *RequestResolutionDate*
  - *CARequestId*

  These certificate search fields parsers have been removed to allow for native EJBCA support in Keyfactor Command as of version 10.0.
- If you have the CA Policy module version 7.0 installed on the same server as the Keyfactor Command Management Portal, you'll need to upgrade the module to version 7.1 or later before running the Keyfactor Command version 10.0 or later upgrade.
- Be Aware of the following certificate store changes in Keyfactor Command v11:
  1. The following store types have been deprecated and will no longer be shipped with Keyfactor Command: All F5 store types, AWS, NetScaler.

     These store types will not be created with new databases. Upon upgrading, they will be removed if the customer does not have any stores or containers defined for the type (per type. i.e. if an AWS store or container is defined and the other types are not, the AWS type will not be removed on upgrade, but the other types will be removed).

2. The built-in functions for IIS and FTP certificate store management in the Keyfactor Universal Orchestrator have been deprecated in Keyfactor Command version 11.0.

   Customers should migrate all uses of IIS certificate store management to the Keyfactor Universal Orchestrator with the IIS custom extension publicly available at:

   https://github.com/Keyfactor/iis-orchestrator

   For more information, see Installing Custom-Built Extensions on page 2940.

   For customers who have FTP certificate store type, when you upgrade Keyfactor Command to version 11 your FTP stores and related data will not be removed on upgrade. In order to manage the FTP certificate store, you will need to use an older version of the UO that still has the FTP extension.

Licensing

You will receive a new license file for the new version of Keyfactor Command. Before upgrading, locate your existing license file so that, should you need to revert to your existing software version, you will easily be able to do so without requesting a new license file from Keyfactor. (License files have the file extension '.cmslicense'.)

As you begin the upgrade, have both your new license file and your existing license file on hand.

If you need assistance with a license, send a request to support@keyfactor.com.

Users, Service Accounts and Groups

Review the Active Directory service accounts and groups used by your Keyfactor Command implementation. You will need to have these accounts and groups available during the upgrade process, along with the passwords for the service accounts. For a full overview of the required service accounts and groups, see Create Service Accounts for Keyfactor Command on page 2757 and Create Groups to Control Access to Keyfactor Command Features on page 2762 in the *Keyfactor Command Server Installation Guide*. The most common service accounts are:

Keyfactor Command Service Account

In many environments, a single service account is used for most Keyfactor Command functions, including the application pool service account and the service account for the Keyfactor Command Service[1]. In some environments, separate service accounts are used for these functions.

Keyfactor Command LogiAnalytics Service Account

Keyfactor Command uses a the reporting engine LogiAnalytics. This reporting engine uses the same service account the application pool is configured to use.

---

[1]If you're running the Certificate Management System rather than Keyfactor Command, this service will be called the CMS Timer Job Service.

Keyfactor Command Orchestrator Service Accounts

If you are using orchestrators, you will need the account(s) the orchestrators are configured to run as and the account(s) used to connect to the Keyfactor Command Orchestrators site.

Keyfactor Command Policy Module

If you are using the Keyfactor Command policy module with any of the standard policy handlers or any custom policy handlers, you will need to have access to upgrade these on the CA if you will be upgrading these at the same time.

CA Gateways

> ⚠ **Important:** All CA Gateways must be upgraded to AnyGateway v22.1 to work with Keyfactor Command v10.

If you are upgrading any of the CA Gateways, you will need to have the correct credentials to connect to the cloud-based certificate authority. The format of these varies depending on the CA provider. Some providers use a username and password while others use client certificate authentication. Some support the choice of either.

If you are unable to locate the existing passwords for your service accounts, you will need to reset the passwords so that the accounts will have known values in preparation for the upgrade. These password changes will need to be coordinated with your existing Keyfactor Command installation to avoid a service interruption. On your Keyfactor Command server(s), the password for the Keyfactor Command service account (assuming you are using just one) will need to be changed:

- In IIS for the CMS/Keyfactor Command application pool.
- In the Services MMC for the Keyfactor Command Service[1].
- Via the Keyfactor Command Configuration Wizard for the LogiAnalytics connectivity.
- Via the Keyfactor Command Orchestrator Configuration Wizard for any orchestrators running in the environment.

Password updates for the Keyfactor Command service accounts can be done via the Keyfactor Command Configuration Wizard during the upgrade process and do not need to be done ahead of the upgrade. The password(s) should be changed in Active Directory as close to upgrade time as possible to limit down time in the existing Keyfactor Command implementation.

If possible, identify the user account that was used to do the original installation of Keyfactor Command (the “installer” account) and use this same account to perform the upgrade. If you are upgrading under a different account than this, the permissions required in SQL will be different. See SQL Permissions on the next page.

[1] Or CMS Timer Job Service for older versions of the software.

SQL Permissions

The user who upgrades Keyfactor Command must have permissions to administer the SQL server and update databases. The user may need to be able to add users (logins), depending on the features used. Full sysadmin permissions in SQL are needed if you're upgrading from a previous version of Keyfactor Command and the user running the install is not the same user who installed the previous version of Keyfactor Command. If the user is the same, only the dbcreator, public and securityadmin roles are needed.

Once Keyfactor Command has been upgraded, these permissions can be removed for the user.

Connecting to SQL over SSL

By default, Keyfactor Command connects to SQL using an encrypted connection. This requires configuration of an SSL certificate on your SQL server.

If your SQL server is not configured correctly for SSL, you'll see an error message similar to the following when you try to make a connection from Keyfactor Command:

```
Unable to establish a connection to the database server. Please ensure that the server
name is correct and sufficient privileges have been granted to the connection account.:
Encountered an invalid or untrusted certificate and could not connect to the database.
TLS encryption is enabled by default. Please visit 'Planning and Preparing --> SQL
Server' In the Keyfactor Installing Server guide to resolve this.
```

To acquire a new SSL certificate or check for an existing certificate, see Using SSL to Connect to SQL Server on page 2746 in the *Keyfactor Command Server Installation Guide*.

If you would prefer not to use an encrypted channel for your connection to SQL, see Configurable SQL Connection Strings on page 2750 in the *Keyfactor Command Server Installation Guide*.

System Requirements

For a full list of the requirements, see System Requirements on page 2702 in the *Keyfactor Command Server Installation Guide*.

Operating System

Keyfactor Command server is supported on Windows Server 2019 or 2022.

PKI Architecture

Please visit Confirm the Architecture on page 3062 and review the implications of upgrading with regard to the PKI architecture elements.

SQL Server

As of Keyfactor Command version 10.0, Microsoft SQL Server 2017, 2019 or 2022 is required and connectivity to the SQL server requires TLS encryption. For information about configuring TLS for SQL server, see Using SSL to Connect to SQL Server on page 2746 in the *Keyfactor Command Server Installation Guide* and:

> https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/enable-encrypted-connections-to-the-database-engine?view=sql-server-ver15

.NET Framework

Microsoft .NET 4.7.2 or greater must be installed on the Keyfactor Command server(s) prior to installation of the latest Keyfactor Command software.

For Windows Server 2019 and Windows Server 2022, .NET is a standard Windows feature added through the Windows Server Manager tool. It can be updated to .NET 4.7.2 or greater with a downloadable update package or through Windows update. For information on checking the .NET version, see Install IIS and .NET on the Keyfactor Command Server on page 2768 in the *Keyfactor Command Server Installation Guide*.

PowerShell Requirement

More recent versions of Keyfactor Command make use of the Active Directory tools for PowerShell to do group membership queries in Active Directory in some functions (e.g. when using a group to create a mapping between a Linux logon for SSH and one or more SSH keys). If this feature is not already installed on your Keyfactor Command server, you will need to install it before upgrading the Keyfactor Command software. The *Active Directory module for Windows PowerShell* is installed as a feature as part of the *Remote Server Administrator Tools*. You may install this through the Roles and Features wizard or using the following PowerShell command:

```
Install-WindowsFeature RSAT-AD-PowerShell
```

Download the Software

Your Keyfactor contact should provide you with a link to download the updated software versions. Be sure to download all the files you will need ahead of the actual upgrade date. This includes the main Keyfactor Command server software as well as the software for the Keyfactor CA Policy Module, any orchestrators (e.g. Keyfactor Universal Orchestrator, Keyfactor Java Agent) or gateways (e.g. AnyGateway), that you will be upgrading at the same time or new software you will be deploying.

Before upgrading to a major version, Keyfactor recommends first upgrading to the final incremental version of the previous major version—completing both the software installation and configuration with the configuration wizard—for the optimal upgrade experience. For example, if you are currently on version 9.4 and want to upgrade to version 10.0 or later, you should upgrade first to version

9.10.1 (the final incremental version of 9.x) before upgrading. Contact your Client Success Manager for more information.

Configuration File

Keyfactor Command can use a file to pass the configuration information into the configuration wizard, which saves a significant amount of typing when you do your initial installation. You may have been provided one of these already configured for your initial implementation, or you may have created one after typing in all the configuration information during the initial implementation. If you can locate this file, it can save some time in the upgrade process. You won't want to import the existing file again, as the file structure may change between versions and importing the file again will overwrite any changes you might have made to the configuration since your initial install, but you can refer to the file for previous configuration information.

The configuration files generally have a .cmscfg extension. When creating the file, you have the option to encrypt and password protect the file. If the file has been password protected, sensitive information in the file, such as any service account passwords, will be encrypted, but the remainder of the file will be human readable. You will need to know the password used to protect the file in order to use the file in its complete state.

Confirm the Architecture

Before you start your upgrade, make sure you have a clear picture of your Keyfactor Command architecture and all the parts that make up the environment, and carefully consider the following.

Roles

Identify all the servers that play a role in the Keyfactor Command environment, including whether you have duplicates of any server roles to support high availability, and make note of what role or roles will need upgrading on each one. Think about whether you want to make any changes to the architecture at this time, such as adding high availability, or consolidating roles.

Certificate Authorities

Keyfactor Command includes a constraint (introduced in version 9.0) that prevents any two certificate authorities from having the same logical name and host name combination. Think about the logical name and host name of the CAs that will be implemented with Keyfactor Command and check for duplicates.

Important: During upgrade, if duplicates are found, then among the duplicates, if there is only one that has any information tied to it, such as certificates, API applications, etc., then all of the others will be removed by the upgrade script. If more than one of the duplicates has any information associated with it, then the upgrade script will stop with an error. In that instance, you will need to manually fix the data before upgrading can proceed.

Templates

Keyfactor Command 10.0 and later upgrades will fail if the database has duplicate templates, defined as:

- Duplicate CommonName and Forest, or
- Duplicate OID and Forest

This should be a rare case. If it does occur, contact Keyfactor support. Support will be able to identify the duplicate templates, save the desired templates, and remove the duplicates.

Backup

Immediately before starting the upgrade, make a backup of these items:

- Your Keyfactor Command SQL database
- Your SQL server Service Master Key (SMK) and/or Database Master Key (DMK), if needed (see Important note)

  If you plan to migrate your Keyfactor Command implementation to a different SQL server during the upgrade, you need a thorough understanding of how Keyfactor Command uses the SMK and DMK. Review the data in SQL Encryption Key Backup on page 821 in the *Keyfactor Command Reference Guide* and make appropriate plans before beginning your upgrade. If you plan to stay on the same SQL instance for the upgrade, you don't necessarily need to backup the SMK or DMK immediately before starting the upgrade. These can just be backed up as part of your normal disaster recovery planning process. Failing to back up the SMK and/or DMK will result in data loss and require manual re-entry of any secret data into Keyfactor Command in the event that the Keyfactor Command database needs to be restored from a backup to a SQL instance other than the original installed instance of SQL server.

  > **Note:** For more information about how Keyfactor Command uses the SMK and DMK and how to back these up, see SQL Encryption Key Backup on page 821 in the *Keyfactor Command Reference Guide*. For more details on the mechanics of SQL Server Encryption and related disaster recovery procedures, see the SQL Server documentation.

- If you're using Keyfactor Command encryption, backup your encryption certificate, with private key (see below).

  More recent versions of Keyfactor Command allow you to encrypt select sensitive data stored in the Keyfactor Command database using a separate encryption methodology. This Keyfactor Command encryption utilizes a Keyfactor-defined certificate on top of the SQL server encryption noted above. This additional layer of encryption (Keyfactor Command encryption) protects the data in cases where the SQL Server master keys cannot be adequately protected. More information is provided in SQL Server on page 2742 in the *Keyfactor Command Server Installation Guide*.
- Backup the NLog configuration file for each application to be upgraded. The location of this file varies depending on the application in question. For older versions of the Keyfactor Command

server, it can be found in one of these locations:

```
C:\Program Files\Common Files\Certified Security Solutions\Certificate Management
System\NLog.config
C:\Program Files\Common Files\Keyfactor\Keyfactor Platform\NLog.config
```

More recent versions of Keyfactor Command separate the NLog configuration into multiple files, with these locations, by default:

```
C:\Program Files\Keyfactor\Keyfactor Platform\Configuration\NLog_
Configuration.config
C:\Program Files\Keyfactor\Keyfactor Platform\KeyfactorAPI\NLog_KeyfactorAPI.config
C:\Program Files\Keyfactor\Keyfactor Platform\Service\NLog_TimerService.config
C:\Program Files\Keyfactor\Keyfactor Platform\WebAgentServices\NLog_
Orchestrators.config
C:\Program Files\Keyfactor\Keyfactor Platform\WebAPI\NLog_ClassicAPI.config
C:\Program Files\Keyfactor\Keyfactor Platform\WebConsole\NLog_Portal.config
```

- Make a backup of the Logi configuration file, which is found here by default:

  ```
  C:\Program Files\Keyfactor\Keyfactor Platform\Logi\_Definitions\_Settings.lgx
  ```

- If you have any custom extension handlers (e.g. auto-registration, alert events), make a backup of these.
- If you've have any other text-based configuration files that have been modified (this is most common for users who have enabled a third-party PAM provider such as CyberArk), make a backup of these.
- If you're using a custom logo for your Management Portal, make a backup of this image. This file can be found here, by default:

  ```
  C:\Program Files\Keyfactor\Keyfactor Platform\WebConsole\Images\Banner.png
  ```

- Review the authentication settings you have configured in IIS for each of the Keyfactor Command applications under the Default Web Site (or other web site if you've installed elsewhere) and make notes as to how they are configured so that you can confirm that the configuration is the same following upgrade.
- If you're using a virtualization solution for your Keyfactor Command application server(s), backup each virtual server as an image.
- If you are using a version of Keyfactor Command older than 6 and have existing SSL scans, export them to a file using the following script. Replace the bold italicized parts with the information relevant to your environment. This step is not necessary if you're upgrading from release version 6 or later.

  ```
  $connectionString = "Data Source=SQLServerName;Integrated
  Security=SSPI;Initial Catalog=KeyfactorDB"
  $connection = new-object
  ```

```
System.Data.SqlClient.SqlConnection($connectionString)
$connection.Open()
# Password can be read from an encrypted file which can be secured as follows:
# Create a password file while logged in as the service account that will run this script:
# $credential = Get-Credential
# $credential.Password | ConvertFrom-SecureString | Set-Content
C:\Keyfactor\PowerShell\encrypted_password1.txt
# use the code below for the credentials
#$password = Get-Content C:\Keyfactor\PowerShell\encrypted_password1.txt | ConvertTo-SecureString
#comment out the below line if using secure credentials
$password = "Password" | ConvertTo-SecureString -AsPlainText -Force
#Update with the credentials for your environment
$username = "domain\administrator"
$credential = New-Object System.Management.Automation.PSCredential($username, $password)
$passphrase = $username + ":" + $password
$fileName = "DiscoveryGroupsExported.txt"
#The name of the agent in your environment
$AgentName = "kyfagent1.domain.com"
#Update with the URLs for your environment.
$kyfAgentUrl = "http://kyfagent1.domain.com/CMSAPI/SSL/1/Agents"
$kyfGroupUrl = "http://kyfagent1.domain.com/CMSAPI/SSL/1/AddEndpointGroup"
$kyfEndpointUrl = "http://kyfagent1.domain.com/CMSAPI/SSL/1/AddEndpoint"
$Bytes = [System.Text.Encoding]::Unicode.GetBytes($passphrase)
$EncodedText = [Convert]::ToBase64String($Bytes)
$headers = @{"Authorization" = "Basic $EncodedText";
       "Content-Type"  = "application/json;" }
$responseAgent = Invoke-RestMethod -Method Get -Uri $kyfAgentUrl -Header
$headers -Credential $credential
if ($responseAgent)
{
  write-host $responseAgent.Name.ToLower()
  if ($responseAgent.Name.ToLower() -eq $AgentName.ToLower())
  {
    $AgentGUID = $responseAgent.Guid
  }
  write-host $AgentGUID
  $kyfEndpointGroups = "http://ky-
fagent1.domain.com/CMSAPI/SSL/1/EndpointGroups?agentId=$AgentGUID"
  write-host $kyfEndpointGroups
  $responseEndpointGroups = Invoke-RestMethod -Method Get -Uri $kyfEndpointGroups -Header
$headers -Credential $credential
  if ($responseEndpointGroups)
  {
    foreach($res in $responseEndpointGroups)
    {
```

```
      write-host $res.Name
      $GroupName = $res.Name
      #write-host $res.guid
      $GroupGuid = $res.Guid
      $sql = "SELECT VALUE, TypeID FROM cms_agents.SslEndpointGroupItems WHERE GroupID =
'$GroupGuid'"
      #write-host $sql $command = new-object System.Data.SqlClient.SqlCommand($sql,$connection)
      $reader = $command.ExecuteReader()
      while ($reader.Read())
      {
        $value = ""
        $type = ""
        $value = $reader["Value"]
        $typeId = $reader["TypeId"]
        #Add-Content $filename "$GroupName,$GroupGuid,$value,$typeId"
      }
      $reader.Close()
    }
  }
}
else
{
  write-host "Agent not found."
}
$connection.Close()
```

The resulting text file will contain the network definitions you currently have and can be opened in Excel. When Keyfactor Command has been upgraded, you can copy and paste from the file into the newly defined *Networks* that replace the previous *Discovery* and *Monitoring* groups.

6.1.2.2 Upgrading

Most Keyfactor Command upgrades are brief with a minimum of changes to existing user accounts, groups, CA templates, firewall settings, etc. The prerequisites have not materially changed from previous versions and the current version can generally be installed using the same hardware and existing instances of the supporting software. The upgrade process is often completed within three to four hours.

Before upgrading, please be sure you have reviewed and addressed the important preparation steps (see Preparing on page 3056).

Important: The MicrosoftECCurveUpgradeModule may fail due to a pre-version 10.0 issue in which Certificate Request Contents were truncated to 4k characters when saved to the database. If your upgrade fails when the MicrosoftECCurveUpgradeModule is run, contact

Keyfactor Support to obtain assistance with the scripts that will have to be run to fix this issue.

Important: During the upgrade process Keyfactor Command prevents duplicate template records from being inserted into the database. Duplicate templates could be found, for example, if there are templates in different forests with the same name. If you receive an error message during upgrade, and the log shows a list of the duplicate templates, contact Keyfactor Support. We will be able to support you through the process of resolving the issue and completing the upgrade.

Before upgrading to a major version, Keyfactor recommends first upgrading to the final incremental version of the previous major version—completing both the software installation and configuration with the configuration wizard—for the optimal upgrade experience. For example, if you are currently on version 9.4 and want to upgrade to version 10.0 or later, you should upgrade first to version 9.10.1 (the final incremental version of 9.x) before upgrading. Contact your Client Success Manager for more information.

The overall task flow consists of the following steps:

Upgrade of the Server Software

In most cases the Keyfactor Command server software can be installed over the existing software installation without uninstalling the previous version. Install the software retaining the same installation location (see Installing on page 2780). In the configuration wizard, populate the fields while referring to your configuration file open in a text editor (see Configuration File on page 3062). Use the existing IIS application pool.

Important: If you're using the Keyfactor CA Policy Module on one or more certificate authorities, you may have been advised in the past to implement a workaround to resolve an error similar to the following:

```
Could not load file or assembly 'CSS.Common, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=0ed89d330114ab09' or one of its dependencies. The system cannot
find the file specified. at CSS.PKI.X509.X509Utilities..cctor()


2022-07-25 17:06:06.6851 CSS.CMS.CA.Msft.PolicyModule.Policy [Debug] -
Certificate request denied or failed by handler
CSSManagedTrustedSubject.PolicyHandler: -2146233088


denied by policy module
```

The workaround involved creating files *CertSrv.exe.config* and *MMC.exe.config* in C:\Windows\System32 on any certificate authority on which you encountered this issue.

With the release of version 11.0 of Keyfactor Command, this workaround needs to be reversed. If the workaround is not reversed, you will encounter errors such as the following on enrollment attempts:

```
Unable to enroll for certificate. The certificate request failed with the reason
'0x80131902.'


2023-10-24 15:57:07.8547 Keyfactor.CA.Msft.PolicyModule.Policy [Error] - Failure
verifying request: Configuration system failed to initialize


Unrecognized configuration section system.web.
(C:\Windows\system32\certsrv.exe.config line 35)
```

To reverse the workaround, on the affected certificate authority:

1. Rename C:\Windows\System32\CertSrv.exe.config to an alternate name or remove it from the System32 directory.
2. Rename C:\Windows\System32\MMC.exe.config to an alternate name or remove it from the System32 directory.
3. Restart the CA services.
4. Confirm that certificate enrollment is working as desired and that the policy handler(s) in place are working as desired.

Upgrade the Keyfactor Universal Orchestrator

In many cases the Keyfactor Universal Orchestrator can be installed over the existing installation without uninstalling the previous version. For specific upgrade guidance, see Upgrading the Universal Orchestrator on page 2895.

Update from Windows Orchestrators

Support for the Keyfactor Windows Orchestrator was deprecated in Keyfactor Command release 11.0. All uses of the Keyfactor Windows Orchestrator should be updated to the Keyfactor Universal Orchestrator. The Keyfactor Universal Orchestrator replaces the Keyfactor Windows Orchestrator and runs on both Windows or Linux servers. The following functions that were part of the Keyfactor Windows Orchestrator are supported in the Keyfactor Universal Orchestrator with custom extensions:

- Interact with F5 devices for certificate management
- Interact with NetScaler devices for certificate management
- Interact with Amazon Web Services (AWS) resources for certificate management
- Interact with Windows certificate stores and IIS

For more information about using custom extensions with the Keyfactor Universal Orchestrator, see Installing Custom-Built Extensions on page 2940.

Important: The Keyfactor Universal Orchestrator is only compatible with Keyfactor Command version 9.0 or later. The current version of the Keyfactor Universal Orchestrator is 11.1 and requires .NET 6.

If you're upgrading from a version of Keyfactor Command prior to 8.0, you will need to update any Windows Orchestrators (a.k.a. Windows Agents) that are used for SSL scanning to support the current scanning architecture. Install and configure the Keyfactor Universal Orchestrator software (see Upgrading the Universal Orchestrator on page 2895).

Note: The orchestrator endpoint location changed for Keyfactor Command release 6 and may need to be modified in your orchestrator endpoint configuration—from CMSAgents to KeyfactorAgents.

Keyfactor CA Policy Module

Important: The most recent versions of the Keyfactor CA Policy Module software need to be upgraded using the below method and PowerShell script and can't be installed over an existing implementation of the Keyfactor CA Policy Module as an upgrade method.

To upgrade a Keyfactor CA Policy Module:

1. Make a note of all your existing policy module configuration, including which policy handlers are enabled and what configurations are set within each handler. **During the upgrade process, you will uninstall the policy module, which will remove your configuration.** The upgrade script should successfully restore the configuration as part of the upgrade process, but you will want to have a complete record of the configuration as a backup.
2. On the Keyfactor CA Policy Module server, open a PowerShell window using the "Run as administrator" option.
3. In the PowerShell window, change to the directory in which you placed the upgrade script included with the latest version of the Keyfactor CA Policy Module and execute it in *archive* mode. For example:

   ```
   .\Keyfactor-CA-Modules-Upgrade-Script.ps1 -Mode archive -InformationAction Continue
   -ErrorAction Stop
   ```

   Note: This step is creating a backup of your policy module configuration before you uninstall the old policy module. It will create an output file, *Keyfactor-CA-Policy.dat*, in the current directory.

Tip: Additional options are available in the upgrade script and can be viewed using the *-full* switch with *Get-Help*. For example:

```
Get-Help .\Keyfactor-CA-Modules-Upgrade-Script.ps1 -full
```

4. Unload the existing policy module in the CA MMC, and close the MMC.
5. **Uninstall the existing policy module**.
6. Install the latest version of the Keyfactor CA Policy Module but do not configure it (see Installing the Keyfactor CA Policy Module Handlers on page 2846 in the *Keyfactor Command Server Installation Guide*). Be sure to install all the same policy handlers that were installed previously.

   Execute the upgrade script included with the latest version of the Keyfactor CA Policy Module again, but this time in *restore* mode. For example:

   ```
   .\Keyfactor-CA-Modules-Upgrade-Script.ps1 -Mode restore -InformationAction Continue
   -ErrorAction Stop
   ```

   Note: This step takes the backup of your policy module configuration from the first run of the upgrade script and restores the information to the correct locations so that you will not need to re-configure the policy module. Be sure that the output file from the first run of the upgrade script, *Keyfactor-CA-Policy.dat*, is in the current directory.

7. Open the CA MMC and load the Keyfactor CA Policy Module (see Installing the Keyfactor CA Policy Module Handlers on page 2846 in the *Keyfactor Command Server Installation Guide*).
8. Open the Properties for the policy module and, if you've received a new license, install the new license on the License tab. On the Custom Handlers tab, review all the configuration to confirm that it has been correctly restored by the upgrade script.

Tip: New versions of the policy module are not necessarily released at the same time as new versions of Keyfactor Command and so the policy module may not need upgrading at the same time as Keyfactor Command.

EJBCA CA Gateway

If you're using an EJBCA gateway and wish to make use of the new feature in Keyfactor Command for native support of EJBCA CAs, you will need to follow the EJBCA gateway upgrade process to unlink the EJBCA certificates in your Keyfactor Command database from your EJBCA gateway CA to enable them to be relinked to a native CA configured in Keyfactor Command. For more information, contact Keyfactor support.

Other CA Gateways

In most cases, the Keyfactor gateway software can be installed over the existing software installation without uninstalling the previous version. Review the configuration for your gateway, and then

install and configure the software as per the Keyfactor gateway guide for the particular gateway, retaining the same installation location. The gateway configuration wizard has significantly changed in recent releases for many of the gateways, which may require modification to your configuration.

Tip: New versions of CA gateways are not necessarily released at the same time as new versions of Keyfactor Command and so gateways may not need upgrading at the same time as Keyfactor Command.

API

Important: The Classic API, also known as the CMS API or CMSAPI, has been deprecated in Keyfactor Command version 11.0. Customers should migrate all uses of the Classic API to the Keyfactor API prior to upgrading to Keyfactor Command version 11.

Please see the latest Release Notes & Upgrading on page 3051 if you are using any custom scripts that leverage the Keyfactor API.

Replacing or Re-Updating Customized Files

Files such as the nlog.config file or customized files for third-party PAM integration (e.g. web.config customizations for CyberArk) may have slight changes in the latest version as compared to the previous version, so you should not just copy your old, customized versions of those files over the current stock versions of these files. You will need to compare the files and make your customizations in the current versions of the files.

Post-Install Configuration and Testing

See Post-Upgrade Steps below

The bulk of the time upgrading will be spent verifying that all functions and configurations have correctly carried over and the upgraded instance is performing correctly.

6.1.2.3 Post-Upgrade Steps

The recommended best practices for after you finish running the Keyfactor Command configuration wizard(s) are:

- The server should be rebooted to assure that the services have a clean start. If this is not possible:
  - Restart Keyfactor Command Service
  - Restart IIS
- Advise users to clear the cache on their web browser and reload the Keyfactor Command Management Portal.

There is no particular order in which the tasks on the following pages must be accomplished.

Tip: If, following the upgrade, you open a page in the Keyfactor Command Management Portal and find it unexpectedly blank or otherwise displaying incorrectly, try refreshing the page with a CTRL-F5. If this doesn't resolve the problem, try clearing the browser cache and then reloading the page. It may be helpful to advise all end users to do this following an upgrade.

Testing

Once everything is up and running again, confirm that the following features are operating correctly:

- Does the Keyfactor Command Management Portal load correctly?
- Run a report in the Keyfactor Command Management Portal to confirm that the connectivity to LogiAnalytics is operating correctly.
- Issue a certificate in the Keyfactor Command Management Portal to confirm connectivity to CAs and that Kerberos authentication is operating correctly (assuming the environment is configured for Kerberos authentication).
- Check the Keyfactor Command log files to confirm that no errors are appearing and that logging is occurring correctly.

Post-Install Configuration

If you are upgrading from any release of Keyfactor Command version 6 or greater, you may want to make some additional configuration changes post-installation:

- Upgrade any Keyfactor CA gateways in your environment that are based on the AnyGateway. The AnyGateway must be upgraded to at least 22.1 to be compatible with Keyfactor Command 10.0 and later.
- Consider whether you wish to implement Keyfactor Command workflows and whether a Keyfactor Command-level workflow could replace CA-level manager approval for any templates that are configured to require CA-level manager approval.

  Note: To prevent REST requests from being made to inappropriate locations by malicious users, if you plan to implement REST type workflows, configure a system environment variable of KEYFACTOR_BLOCKED_OUTBOUND_IPS on your Keyfactor Command server pointing to the IP address or range of addresses in CIDR format that you wish to block. Both IPv4 and IPv6 addresses are supported. More than one address or range may be specified in a comma-delimited list. For example:

  ```
  192.168.12.0/24,192.168.14.22/24
  ```

  When a REST request is made where the URL is either configured to a blocked IP address or resolves via DNS to a blocked IP address, the REST request will fail.

- Review the new enrollment default and policy settings for enrollment. Enrollment defaults and polices can be defined at two levels:

- System-wide settings apply to all enrollments done through Keyfactor Command unless they are overridden by template-specific settings.
- Template-level settings allow you to modify any established enrollment defaults or policies on a per-template basis.

There are several settings available for configuration as part of the template policies:

- Allow Wildcards
- Allow Public Key Reuse
- Enforce RFC 2818 Compliance
- Supported Key Types

Enrollment defaults allow you to pre-polulate the subject fields in PFX Enrollment and CSR Generation. Users are allowed to override these at enrollment.

- If you're using certificate metadata or regular expressions, optionally define these for each template. Certificate metadata fields and regular expressions can be defined at two levels:
  - System-wide settings apply to all enrollments done through Keyfactor Command unless they are overridden by template-specific settings.
  - Template-level settings allow you to modify any established certificate metadata or regular expressions on a per-template basis (for instance, for a metadata field, whether the field is required, what default value it should provide, or whether to hide the field during enrollment, regardless of system-wide setting).
- Review any alert PowerShell event handlers you may have configured to ensure they are in the path (or subdirectory thereof) as defined in the *Extension Handler Path* application setting value. Changes as of version 9.0 will cause PowerShell event handlers to fail if not located in the defined directory. For more information, see Adding PowerShell Handlers to Alerts on page 223 in the *Keyfactor Command Reference Guide*.
- The enrollment configuration will have been carried over in the upgrade, however you may want to confirm the configuration of Certificate Authority and Template enrollment (PFX, CSR, and CSR generation) and make any changes.
- Review any template that is configured to require manager approval at the CA level and confirm that a Keyfactor Command private key retention policy in place.
- Update any monitoring or other processes that reference the log files to point to the new log file location.
- Review the new reports in the Keyfactor Command Report Manager and add them to the menu or favorite them, if desired.

If you are upgrading from a release prior to Keyfactor Command version 6.1, please contact support (support@keyfactor.com) for upgrade assistance.

6.1.2.4 Troubleshooting

Typically, an upgrade completes with few hiccups and the new version of Keyfactor Command comes up without incident. If this doesn't happen, start by checking the log file(s) for any errors. By default, these are located in C:\Keyfactor\logs. It is sometimes helpful to enable debug or trace level

logging. This is done by editing the nlog.config file for each application. For more information, see Editing NLog on page 796 in the *Keyfactor Command Reference Guide*.

Error During Upgrade

If you encounter an error during upgrade, this can be the result of a number of different things. Often, it's related to connectivity to SQL or issues on the SQL server. Check the *Command_Configuration_Log.txt* for messages related to upgrading and upgrade failures. This will point you in the right direction to begin troubleshooting.

The following error message indicates that the referenced upgrade script failed because it took longer to run than the allowed limit for SQL tasks:

```
2022-12-07 10:19:07.5078 Keyfactor.Sql.Management.Upgrade.UpgradePlan [Error] - Failed
to run upgrade module CSS.CMS.Install.Upgrade.Scripts.EJBCA_Resolved_Request_Contents_
Removal.sql: Execution Timeout Expired. The timeout period elapsed prior to completion
of the operation or the server is not responding.
```

The Keyfactor Command upgrade process includes multiple scripts, each doing different tasks, and each script is run in batches to limit the time and load of any one SQL request, but it's still possible to encounter a batch that exceeds the limit with vary large or complex databases. To resolve this particular issue, you can increase the timeout limit and restart the upgrade. You do not need to restore and start over.

To increase the timeout limit:

1. On the Keyfactor Command server, open a text editor (e.g. Notepad) using the "Run as administrator" option.
2. In the text editor, browse to open the *CSS.CMS.Install.ConfigurationWizard.exe.config* file (*ConfigurationWizardConsole.exe.config* if you're doing a command-line install) in the Configuration directory under the installed directory. By default, this is:

   ```
   C:\Program Files\Keyfactor\Keyfactor
   Platform\Configuration\CSS.CMS.Install.ConfigurationWizard.exe.config
   ```

3. Locate the appSettings line that contains *Keyfactor.Sql.DbCommandTimeout*. This will look something like:

   ```
   <add key="Keyfactor.Sql.DbCommandTimeout" value="1800" />
   ```

4. The timeout value is set in seconds, so the default of 1800 seconds is 30 minutes. Set it to a new, longer value to allow the upgrade to complete. Don't set it to a value that's too high, as you do want the upgrade to time out if there's some fundamental problem communicating with SQL.
5. Save the file, close the configuration wizard, open the configuration wizard again (you should find it on the menu), and begin the upgrade again.

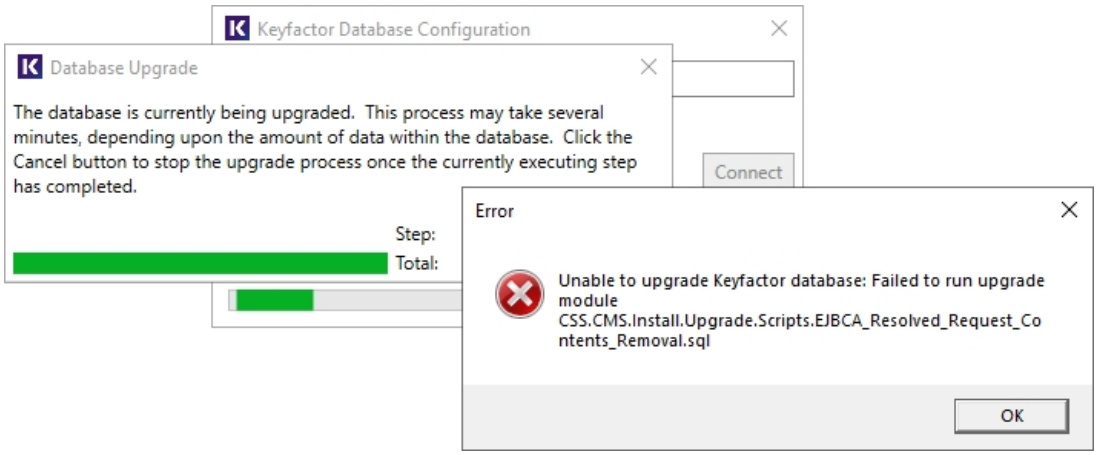


Figure 621: Error During Upgrade

Note: In previous versions of Keyfactor Command, the timeout was controlled with the *command timeout* setting in the connection string of the *SharedSqlConnectionStrings.config* file and had a default of 360 seconds (6 minutes).

Management Portal Doesn't Load After Upgrade

If the Keyfactor Command Management Portal appears to partially load or does not appear to include expected updates after the upgrade, try clearing the browser cache, closing the browser, and opening a fresh browser session. Try using CTRL-F5 to request the page again without cached content. In some upgrade cases, with Internet Explorer, the Certificate Search page only partially loads. With some browsers, opening the Developer Tools with the F12 key and clearing the cache will resolve the problem.

Certificate Enrollment Fails

If the certificate enrollment fails, this is often an indication that there is a Kerberos authentication problem. Confirm that the service principal name (SPN) is set correctly for the application pool service account and that Kerberos constrained delegation is configured correctly from the Keyfactor Command server(s) to the CA(s). For more information, see Configure Kerberos Authentication on page 2822 in the *Keyfactor Command Server Installation Guide*.

Event Handlers Don't Run

If your alert PowerShell event handlers or renewal event handlers do not run correctly, be sure that you have updated them to the correct new location. Scripted alert handlers will fail to run if not in the path (or a subdirectory of it) specified by the *Extension Handler Path* application setting. By default, this is C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\. For more information, see Adding PowerShell Handlers to Alerts on page 223 in the *Keyfactor Command Reference Guide*.

500 Error on the Dashboard or in Reports

If you receive a 500 error loading the dashboard and running reports but the remainder of the Management Portal seems to be operating correctly, check to be sure that the IP address(es) configured in the Configuration Wizard on the Dashboard and Reports tab have been entered correctly.

Underlying Connection Closed

If you receive an error when opening the Management Portal that "the underlying connection was closed" please be sure you have all the latest Windows updates installed.

Please refer to the Keyfactor Command Release Notes & Upgrading on page 3051 for known issues.

If you need further assistance, please contact support. During normal business hours, support can be reached at support@keyfactor.com or (877)-715-5448.

6.2 Major Release 11.0 Notes

October 2023

We're thrilled to announce Keyfactor Command 11.0, which includes some major new features and updates such as support for OAuth 2.0, Universal Orchestrator Container, and certificate renewal tracking.

Please refer to Upgrade Overviews on page 3051 for important information about the upgrade process.

Important: The MicrosoftECCurveUpgradeModule may fail due to a pre-v10 issue in which Certificate Request Contents were truncated to 4k characters when saved to the database. If your upgrade fails when the MicrosoftECCurveUpgradeModule is run, contact Keyfactor Support to obtain assistance with the scripts that will have to be run to fix this issue.

Tip: We encourage customers to contact their customer success manager to discuss the new features and functionality in Keyfactor Command 11, and to schedule an upgrade. Please refer to Upgrade Overviews on page 3051 for important information about the upgrade process.

Highlights

New Security Permission Model

For release 11.0 of Keyfactor Command, a new permission structure has been introduced. Users of Keyfactor Command through the Management Portal will see some changes, including:

- The permissions tabs on the Security Roles Role Information For: *Role Name* page have a new look and method for setting permissions using check boxes in a tree structure.
- The Security Identities page has been removed.
- A new page for Claims has been added. This page allows the user to see the roles a claim is in.
- The security permissions for PKI Administration (*PKI Management: Read* and *PKI Management: Modify*) have been replaced with two new security permissions: *Certificate Authorities* (*Read* and *Modify*) and *Certificate Templates* (*Read* and *Modify*). The upgrade process will appropriately translate the old security permission to the new ones.

Users of Keyfactor Command through the Keyfactor API will need to understand the new model.

On upgrade, existing security roles will be converted to the new permission structure.

The version one permission model was largely replaced in Keyfactor Command version 11.0, but is retained for backwards compatibility for use with select Keyfactor API endpoints.

Support for OAuth 2.0

Release 11.0 of Keyfactor Command introduces support for open authorization (OAuth) 2.0 compliant identity providers as an alternative to Microsoft Active Directory. Keyfactor offers the Keyfactor Identity Provider, which is a lightweight application that is easily installed in the same environment as Keyfactor Command and can be used to provide standalone authentication with user and group management for Keyfactor Command. It may be used directly or federated to another open authorization (OAuth) 2.0 compliant identity provider (e.g. Okta, Ping Identity). Keyfactor Identity Provider runs in a Linux-based container (e.g. Docker).

The Management Portal has been updated to include a new system setting, *Identity Providers*, that provides the ability to view and edit the identity provider configured for your Keyfactor Command implementation. Active Directory as an identity provider cannot be edited. New identity providers should be added through the Keyfactor Command Configuration Wizard. Authentication with only one identity provider at a time for a given Keyfactor Command server is supported.

When using Keyfactor Identity Provider, there is now a login page to present the user with a login prompt where he or she can choose to login directly to Keyfactor Identity Provider or click a link that will redirect to the login page of the federated identity provider.

The logout functionality for users authenticated with Keyfactor Identity Provider will log the user out and redirect them to the login page. Users authenticated via Active Directory will receive a message instructing them to “Close your browser to sign out” as per earlier versions of Keyfactor Command.

Identity provider secrets can be managed by a PAM provider, if desired. Please see the Identity Provider Operations on page 755 for more information.

Important: Select substitutable special text tokens provide the ability for parts of the Keyfactor Command system to look up data in Active Directory to use in alerts and workflows.

⚠ Lookups for these locate the object in Active Directory identified by the user or computer account that requested the certificate from the CA and substitute the contents of the referenced attribute. For example, an alert might reference *requester:displayname* to look up the requester's display name in Active Directory.
The substitutable special text tokens that begin *requester:* or *principal:* (but not the standalone *requester* token) are only supported when using Active Directory as an identity provider. They cannot be used with other identity providers since there is no Active Directory to query. Customers wishing to use similar functionality with a different identity provider will need to do API calls to query the identity provider, populate the data into a metadata field, and use the metadata field when populating alert and workflow messages.

Certificate Renewal Tracking

When certificates are renewed in Keyfactor Command via the Management Portal or Keyfactor API, a new *Keyfactor Renewal* record is created. This can be used as a filter for *Ignore Renewed Certificates* in certificate collections. A history record will now show on the original and renewed certificates showing the thumbprint it was renewed from/to.

Keyfactor Universal Orchestrator in a Container

The Keyfactor Universal Orchestrator can now be deployed as a Linux container using a containerization platform such as Docker or Kubernetes. This will make deployments faster and less error prone and can fit into the CI/CD pipeline.

Updates

Changes & Improvements

- **Script/Handler Library in Command DB**
  - POST Extension/Script API endpoint added for registering PowerShell scripts for alerts for certificate requests, SSH and workflow, that results in their storage in the Keyfactor Command database (there is no Management Portal equivalent for security reasons). Alert and custom workflow scripts are now loaded from the Keyfactor Command database instead of the file system and available from dropdowns on any dialogs that use scripts, so users can select from the list of registered scripts. The upgrade process will retrieve scripts from the extension library folder and import into Keyfactor Command DB.
  - When configuring alerts and workflows, script selection now appears as a dropdown in the Management Portal. Only those scripts that have been defined in the Keyfactor Command database and configured for the selected category appear in the dropdown.
  - When upgrading from a version of Keyfactor Command prior to version 11, the upgrade process will search the file location defined in the *Application Settings > Console Tab > Extension Handler Path* setting and add all the files found in that directory to the database with the naming convention of *foldername (_subfolder name, if applicable)_filenname* so it is

clear which scripts were imported from which location. The upgrade process will also identify which, if any, of the categories the script is configured for and add that information to the database with the script so that existing scripts will work upon upgrade. The extension library directory has an added file called *Migration-<date>.txt* that informs the user that this directory has had its scripts migrated to the database.

- **.NET Updates**

  Updates have been made to allow the Keyfactor Command solution to multi target to .NET Framework (NET FX) 4.7.2 and .NET Core 6.0 to allow progress towards the goal of allowing Keyfactor Command to run on Linux-based operating systems and achieving cross-platform containerization. Eventually, the platform will not use NET FX.

- **System Alert Bar Design Change**

  System alerts are now indicated by a bell icon in the header. Clicking on the bell will open a menu of alerts:

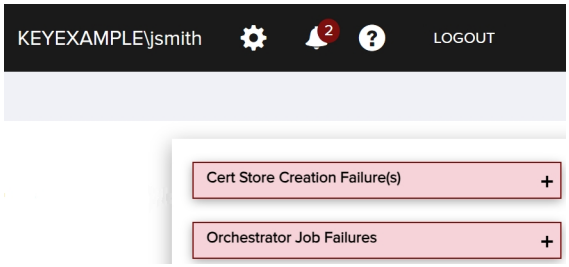
  

  Figure 622: System Alerts

- **One-Click Renewal Enhancements**

  More control was added for allowing one-click renewal on certificates. If you wish to use *One-Click Renewal* for certificates, the **Allow One-Click Renewals** option must be enabled in both the templates and CAs to which you want *One-Click Renewal* to apply (see Certificate Template Operations on page 381 and Adding or Modifying a CA Record on page 354). For more information about one-click renewals, see Renew on page 69.

- **Enrollment Enhancements**

  Added an information message on the PFX Enrollment page to let the user know the required password length.

  Added three new download options to PFX Enrollment: PEM, DER, and P7B.

  Added the option to select an algorithm and key size when using CSR Generation and PFX Enrollment.

  Added an auto-select option to the enrollment pages. When a template is selected that has 2 or more available CAs, the default option in the CA drop down will be auto-select, which will use the first eligible and reachable CA to do the enrollment.

  PFX Enrollment, CSR Enrollment, and CSR Generation now support RSA key size 3072.

- **New Email Metadata Type**
  For new databases the standard *Email-Contact* metadata field will be created as the new email metadata type field. For existing databases, any *Email-Contact* data will be unaffected, but the new metadata type will be available.
- **Alerts**

  Expiration alerts will no longer determine the time range based on the *LastExpiration* service settings field. Instead, each expiration alert will have its own last execution time. When created, the value will be set to the current UTC time. The next time it is executed, the time span will be from the time in the LastExecution field and the current UTC time. After execution, the time will update to the current UTC time. Previously, alerts would update the LastRun field in the service settings. This no longer happens. The LastRun field was removed. Users can still see when alerts were last run by looking at the event logs.
- **Installation and Configuration Changes**

  The configuration wizard service tab has been changed. Individual Keyfactor Command Service job check boxes have been removed, leaving only the **Start service on bootup** checkbox. By checking **Start service on bootup** all jobs default to **true**. A given job can be disabled by changing this setting to **false** in the appsettings.json file (see Keyfactor Command Service Job Settings on page 778 for more information).

  The configuration wizard settings *{filename}.cmsfg* file in Keyfactor Command version 11 has a new <AdminUsers> section, where the settings for the **Administrative Users** tab of the configuration wizard are saved/defined. This new section contains the additional parameters for the new identity provider and security protocol in v11. In previous Keyfactor Command versions, the **Administrative Users** were saved/defined in the *<Console>* section, *<AdminUser>* parameter. Running the configuration wizard for v11 with an older version *{filename}.cmsfg* will populate the configuration wizard **Administrative Users** tab from the details in the *<Console>* section, *<AdminUser>* parameter. Once populated, additional information will need to be provided on the **Administrative Users** section for v11.
- **Supported Browsers**

  The supported browsers for Keyfactor Command have been updated to:
  - Chrome: 99.0.4844.74
  - Firefox: 98.0
  - Microsoft Edge: 99.0.1150.30
- **Certificate Search Updates**

  Added download options to the Certificate Search page. The options are now: PEM, DER, P7B, PFX, ZIP PEM, and JKS. (Previously, the options were PEM, SER, P7B, and PFX.)

  Users without renew certificate permission will no longer see the Renew button.

  On the Certificate Store - View Inventory dialog when you click **Query Certificate** on a selected inventory item, Keyfactor Command will include the *Revoked* and *Expired* check boxes if either condition is true for that certificate.

When a certificate collection is saved, the collection record will include the *Revoked* and *Expired* check box state and these will load in the appropriate state when the collection is loaded.

- **Certificate Templates**

  On the Certificate Templates grid, the Key Type column is renamed to Key Types and displays all the template's key algorithms and key sizes in a comma-separated list with format *KeyType KeySize*. For example, a template that supports all key types would look like this: RSA 2048, RSA 4096 and ECC p-256, ECC p-384. Currently, the supported key types are RSA, ECC, ECDSA, Ed448, Ed25519. There is also a new section *Key Types and Sizes* on the *Template > Details tab* which list all key type and size information for the template.

  Support is now provided in Keyfactor Command for a single certificate template containing multiple key types, sizes and/or curves.

- **Certificate Store Types**

  Certificate Store Type Entry Parameters can now be added, edited, or deleted regardless of whether the certificate store type is in use or not. It should be noted that the parameter type (String, Bool, Multiple Choice, or Secret) cannot be changed.

  Certificate Store Type Custom Fields can be added, edited, and deleted even if the type is in use. The custom field's type, however, cannot be changed after it is created. If you want to change that, you must delete the property and add it back with a different type.

  Certificate Store Type Short Name and Name fields can be edited. The Custom Capability field length has been expanded to 64 characters.

- **Keyfactor Command Service**

  New Keyfactor Command Service related application settings have been added around lock acquisition for ensuring that only one Keyfactor Command Service job runs at a time across multiple servers. On the Console > General tab:

  - Lock Timeout (seconds) - The amount of time to attempt to acquire a lock.
  - Lock Heartbeat Interval (seconds) - How often to update the lock to keep it alive while running a long running Keyfactor Command Service job.
  - LockHoldTimeout (seconds) - How long to wait after the last successful heartbeat interval before the lock is considered to be lost and can be acquired by another machine.

- **Certificate Stores**

  The maximum length for certificate store reenrollment jobs has been increased from 256 chars to 850 chars.

- **Custom Extension Handling Updates**

  To reduce the number of assemblies listed in the shared assemblies file, a new extension registration option, *LoadInUpstreamContext*, is available. When set to *True* it causes all of the extension's assemblies to be shared with the calling assembly. The *SendEmailOrchestratorJobCompleteHandler* should be updated to add and set *LoadInUpstreamContext* to *True* (see Editing Job Completion Handlers on page 771).

- **Logging**

  Improved logging for EJBCA Gateway configuration connection errors.

  Correlation IDs are now returned in the API 's HTTP response header as *X-Keyfactor-Correlation-Id*. These match the correlation GUID tracked in the log files.

- **Improved CA Synchronization Performance**

  Improved performance for synchronizing CAs that have a large number of duplicate keys.

- **Dashboard**

  The ability to authenticate to the Dashboards and Reports Engine, Logi Analytics, no longer requires IP addresses.

  The error message that displays in the case of an error condition has been simplified to remove any sensitive information that might reveal inappropriate information to an attacker (e.g. server hostname or IP address). The message will be displayed as: *Your logon credentials failed. Please contact Keyfactor support at support@keyfactor.com for assistance with report customization or migration to the new platform.*

- **SQL Updates**

  Previously, SQL retry logic was controlled by parameters in the web.config files. With the migration to .NET 6, these settings have been moved to the appsettings.json file located in the Configuration folder of the KeyfactorAPI and/or WebConsole folders. Additionally, to allow for the rest of the product to also use the same retry logic, the parameters in the web.config files for projects that are still in .NET framework have been moved to the appsettings section of the web.config. See Keyfactor Command Changing SQL Retry Settings on page 787 for more information.

- **Orchestrator Management**

  The WebAgentServices\Configuration\appsettings.json file has some added configuration settings for managing orchestrator behavior. For more information, see Keyfactor Command Web Agent Services on page 773

  For Keyfactor Command instances configured to use Keyfactor Identity Provider, the Keyfactor Orchestrator API calls support the use of bearer tokens.

- **PAM**

  Added an optional capability to the Universal Orchestrator to ignore certificate store passwords from Keyfactor Command and obtain them from a client-side PAM provider instead. For more information, see Installing Custom PAM Provider Extensions on page 743.

  In previous versions of Keyfactor Command, once a PAM provider had been associated with an orchestrator whose certificate stores were configured to retrieve their secrets from a PAM provider, it was not possible to edit the PAM provider configuration. In Keyfactor Command v11, the PAM configuration can be edited even if the configuration is in use. Note that the PAM configuration can not be deleted if it is in use.

  PAM provider role based access controls have been updated. Previously, certificate store containers were used to provide the security for PAM providers. Since PAM is now being

leveraged across Keyfactor Command, this has been removed and replaced with a new security role permission for PAM providers.

- **CA Policy Module**

  When upgrading from a previous version of the Keyfactor CA Policy Module, refer to Keyfactor CA Policy Module on page 3069 for important upgrade guidance and an upgrade script.

- **Workflow**

  Added a new REST API step in the workflows to support the use of an OAuth bearer token.

  In addition to the tokens in the dropdown in Workflows, any data in the current data bucket can be referenced by entering an appropriate reference string. For example, to return the CSR for an enrollment request you can use $(CSR). Refer to the CurrentStateData field in the response to the GET /Workflow/Instances/{instanceId} API method for information on all the data found in the current (as opposed to initial) data bucket (see GET Workflow Instances Instance ID on page 2630).

  The PowerShell *Set Variable Data* workflow step now allows support for the ConvertFrom and ConvertTo cmdlets.

Fixes

- Denied alerts configured with metadata fields as substitutable special text tokens in the message were not getting the metadata values inserted on alert generation.
- The GET /OrchestratorJobs/JobStatus/Data endpoint was returning a 404 error even with a valid jobHistoryId.
- Template names were being treated as case sensitive during enrollment API calls. In addition resolution this issue, logging improvements were made to assist in troubleshooting similar items in the future.
- The CSR generation page would not allow input of subject information if the RFC 2818 compliance policy was turned on at either the template or system-wide settings level.
- Deleting a CA and then re-adding it was causing the CA sync to get stuck.
- OCSP files could not be uploaded in the Management Portal.
- Editing certificate metadata caused the certificate import date to change.
- Metadata fields could not be deleted if they had custom settings defined on a template.
- Pending certificate requests without CNs could not be approved.
- The dashboard was not accurately reflecting the counts of certificate collections that included the revoked and expired certificates.
- The certificate store type “Depends On” radio button would not display the available fields when selected.
- The Metadata, Enrollment RegExes, and Enrollment Defaults tabs on Certificate Templates were incorrectly displaying multi-select checkboxes.
- The Dashboard could hang when trying to remove a widget.

- An LDAP CRL could hang when loading the Dashboard Revocation Monitoring widget.
- Reports that have templates as parameters were not allowing the template parameter to be selected.
- When saving an HTTPS CA, on the Authorization methods tab, there is a database field length limit that applies to the secret supplied for the authentication certificate. Errors encountered with a lengthy secret for the certificate have been resolved by saving only the end-entity certificate to the database, rather than the whole chain.
- Certificates that were part of a certificate renewal chain could not be deleted. Now, whenever a certificate is deleted that is a part of a certificate renewal chain, the certificates on either end of the deleted certificate(s) will have their certificate histories updated to show that either a certificate before or after the certificate was deleted in the renewal chain of that certificate.
- Prior to Command v11, it was possible to create a Keyfactor schedule via the API that was impossible to display via the Management Portal schedule picker. For example, an interval that is not on the interval dropdown (e.g., every 107 minutes). Now, the API will also not allow interval values to be saved that cannot be represented in the Management Portal. Previously saved schedules that don't meet these restrictions can still be parsed and used, but not edited and re-saved, as a result errors may be encountered on re-saving invalid schedules that were created via the API.

Deprecation

> **Important:** The Classic API, also known as the CMS API or CMSAPI, has been deprecated in Keyfactor Command version 11.0. Customers should migrate all uses of the Classic API to the Keyfactor API prior to upgrading to Keyfactor Command version 11.

- The Keyfactor Java Agent will be deprecated in a future release of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at: https://github.com/Keyfactor/remote-file-orchestrator.

  For more information, see Installing Custom-Built Extensions on page 2940.
- The following store types have been deprecated and will no longer be shipped with Keyfactor Command:
  - All F5 store types
  - AWS
  - NetScaler
  - IIS certificate store types

  These store types will not be created with new databases. Upon upgrading, they will be removed if the customer does not have any stores or containers defined for the type (per type. i.e. if an AWS store or container is defined and the other types are not, the AWS type will not be removed on upgrade, but the other types will be removed). If you would like to manage these types of

stores, you will create new certificate store types as per the appropriate Keyfactor-built custom extension for managing the selected store type (see Installing Custom-Built Extensions on page 2940).

- The built-in functions for IIS and FTP certificate store management in the Keyfactor Universal Orchestrator have been deprecated in Keyfactor Command version 11.0.

  Customers should migrate all uses of IIS certificate store management to the Keyfactor Universal Orchestrator with the IIS custom extension publicly available at: https://-github.com/Keyfactor/iis-orchestrator.

  For more information, see Installing Custom-Built Extensions on page 2940.

  For customers who have FTP certificate store type, when you upgrade Keyfactor Command to version 11 your FTP stores and related data will not be removed on upgrade. In order to manage the FTP certificate store, you will need to use an older version of the UO that still has the FTP extension.
- The prescript and postscript functionality of the Keyfactor Universal Orchestrator has been replaced by other functionality in Keyfactor Command such as that provided by Keyfactor Command workflows (see Workflow Definitions on page 230). As a result, prescript and post-script functionality has been deprecated and will be removed from a future release.
- Windows Orchestrator has been deprecated in v11. The reason for this change is that there has not been a new version since Keyfactor Command v8 and our policy is to support orchestrators for the latest two (2) versions of the products. Additionally, all Windows Orchestrator func-tionality is available in the new Universal Orchestrator and it's associated extensions.
- In preparation for separating the vSCEP web application into its own solution, the configuration wizard no longer provides the option to install vSCEP. Customers wishing to upgrade and also still use vSCEP should install vSCEP on a separate server. The separate vSCEP application will be released at some point in the future.

Future Changes

Known Issues/Limitations

- Updating a workflow step's type without changing the unique name will result in an error. To get around this issue, when changing a workflow step's type also change the step's unique name. Another work-around is to delete the step you no longer want and add the new step you do want.
- If the .NET Hosting Bundle is installed before IIS is installed, you may have to repair the hosting bundle after IIS is installed. The official Microsoft page for the hosting bundle specifically mentions this scenario is at: https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/hosting-bundle?view=aspnetcore-7.0#install-the-net-core-hosting-bundle.
- Although permissions for a container can be viewed or modified using the permission option on the certificate store containers tab, Keyfactor recommends best practice is to manage permis-sions as part of the overall permission configuration (see Security Roles and Claims on page 622), as additional system-wide configuration settings are available there that cannot be viewed or modified from certificate store containers - permissions page.

- Any renewal instances that were in-flight within the workflows that have been started in version 10 will not create a new renewal link in the certificate's renewal chain unless the workflow instance was created after an upgrade.
- In Keyfactor Command v11, the v1 security role endpoints only work for Active Directory users. If you move to OAuth and have any integrations against certain security roles APIs, they will not work. Security API endpoints that support OAuth have been added as version v2 of the endpoints.
- The Management Portal and the Keyfactor API do not display duplicate SANs. For certificates that have duplicate SANs, the Management Portal and Keyfactor API only show the unique list and count of the certificate SANs. For example, a certificate with four SANs, of which three are the same (duplicates), would only show two unique SANS on the Certificate Details page and the GET Certificates Keyfactor API method.
- SSH management in Keyfactor Command with the Keyfactor Bash Orchestrator is only supported when using Active Directory as an identity provider (see Selecting an Identity Provider for Keyfactor Command on page 2704). The SSH option in the Management Portal will only appear when Keyfactor Command is installed using Active Directory as an identity provider (and with a license that supports SSH).
- Orchestrator auto-registration in Keyfactor Command is only supported when using Active Directory as an identity provider (see Selecting an Identity Provider for Keyfactor Command on page 2704). If you need auto-registration with Keyfactor Identity Provider authentication, see Custom Auto-Registration Handlers on page 495.
- The configuration options on the Keyfactor Universal Orchestrator for remote CA management require that the *AdditionalCertificateAuthoritiesAllowed* value be set to **true** regardless of whether you populate the *CertificateAuthorities* section with your CA information.
- In version 11, if you attempt to access the Management Portal when the SQL server is offline, you will get a blank page with a 503 unavailable error. Note that an IIS reset will be needed to connect to the Management Portal when you restart the SQL server. In version 10 and prior, you would get a screen that said that there was an issue.
- When using Keyfactor Command on a non-domain-joined server, there are some areas in the Keyfactor Command Management Portal that may cause confusion since they reference Active Directory. These places are not going to be addressed for 11:
  - Certificate authority import may show an empty dropdown.
  - The orchestrator auto-registration Validate Users function will return a *could not be resolved* error.
- When querying a workflow by ID with the *contains* or *starts with* operators and a partial id value an error is reported that it is an *Invalid Id value*.
- In the configuration wizard under Authentication, there is a field for *Identity Provider Hint*. This field has been deprecated and will be removed in a future version of Keyfactor Command.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

To reflect the changes made during the .NET 6 migration, in Keyfactor Command version 11, the *Keyfactor API Reference and Utility* (Swagger) link from the Management Portal is changed as follows:

- The URL for the *Keyfactor API Reference and Utility* changed from https:/*keyfactor.keyexample.com*/**KeyfactorAPI**/ref/index#

  to https://*keyfactor.keyexample.com*/**KeyfactorAPI**/swagger/index.html

  Tip: Where *keyfactor.keyexample.com* is the Keyfactor Command Management Portal server and **KeyfactorAPI** is the virtual directory assigned to the Keyfactor Command Management Portal in the configuration wizard.

- There is no longer an individual URL link to each API endpoint within the *Keyfactor API Reference and Utility* (such as *https://keyfactor.keyexample.com/KeyfactorAPI/ref/index#!/Certificate/Certificate_QueryCertificates*) at the bottom of each *Keyfactor API Reference Guide* documentation page—just the one link to the main page of the *Keyfactor API Reference and Utility*.
- At the top of the *Keyfactor API Reference and Utility* under the title there is a link to allow easy access to the raw OpenAPI JSON.

Important: The Classic API, also known as the CMS API, was deprecated in Keyfactor Command version 11. All uses of the Classic API should be migrated to the Keyfactor API prior to upgrading to Keyfactor Command version 11.

Table 856: API Change Log

Endpoint	Methods	Action	Notes
AppSetting	GET, PUT	Added	
AppSetting/{id}	GET	Added	
AppSetting/{id}/Set	PUT	Added	
AppSetting/{name}/Set	PUT	Added	
CertificateAuthority/SourceCount	GET	Added	
CertificateAuthority/ConfigurationTenants	GET	Added	
CertificateAuthority/HealthMonitoring/Schedule	GET	Added	

Endpoint	Methods	Action	Notes
CertificateAuthority/AlertRecipients/CAHealthRecipients	GET	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients	POST	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients	GET	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients	POST	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients/{id}	DELETE	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients/{id}	GET	Added	
CertificateAuthority/AlertRecipients/CAHealthRecipients/{id}	PUT	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id}	DELETE	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id}	GET	Added	
CertificateAuthority/AlertRecipients/CAThresholdRecipients/{id}	PUT	Added	

Endpoint	Methods	Action	Notes
CertificateAuthority/Import	POST	Added	
CertificateAuthority/ConfigurationTenants	GET	Changed	The endpoint is now renamed to GET /CertificateAuthority/AvailableForests and the definition is changed to: Returns a list of available forests that are in Active Directory.
Certificates/CSV	GET	Added	
Certificates/IdentityAudit/{id}	GET	Added to V2 definitions	This API endpoint is available in both the V1 and V2 definitions in the Keyfactor API Reference and Utility and acts exactly the same in both.
CertificateCollections/{id}/Permissions	POST	Removed	Instead use POST Security/Roles/{id}/Permissions/Collection.
CertificateCollections/{id}	DELETE	Added	
CertificateCollections/NavItems	GET	Added	
CertificateCollections/CollectionList	GET	Added	
CertificateCollections/{id}/Favorite	PUT	Added	
CertificateStores/Server	GET, POST, PUT	Deprecated	
CertificateStoreTypes	GET	Changed	The API will return ALL certificate store types if at least one of these conditions are met: • The end-user has one of the /certificate_

Endpoint	Method-s	Action	Notes
			stores/read/ global permissions. • The end-user has permission to at least one certificate store container.
ComponentInstallation/{id}	DELETE	Added	
ComponentInstallation/	GET	Added	
EventHandlerRegistration/{id}	GET, DELETE, PUT	Added	
EventHandlerRegistration/	GET, POST	Added	
Extensions/Scripts/{id}	DELETE, GET	Added	
Extensions/Scripts	GET, POST, PUT	Added	
IdentityProviders/{id}	GET, PUT	Added	
IdentityProviders	GET	Added	
IdentityProviders/Types	GET	Added	
Permissions	GET	Added	
PermissionSets/{id}	GET, DELETE	Added	
PermissionSets	GET, POST, PUT	Added	
Scheduling	POST	Added	
Security/Containers/{id}/Roles	GET,	Added	

Endpoint	Methods	Action	Notes
	POST		
Security/Audit/Collections/{id}	GET	Added	
Security/Claims/{id}	GET, DELETE	Added	
Security/Claims	GET, POST, PUT	Added	
Security/Claims/Roles	GET	Added	
Security/Identities	GET	Changed	The non-working query string field has been removed.
Security/Roles/{id}/Permissions/PamProviders	GET, PUT	Added	
Security/Roles (V1) Security/Roles/{id} (V1) Security/Roles/{id}/Identities(V1) Security/Roles/{id}/copy(V1)	GET, POST, PUT	Deprecated in V1	All SecurityRoles API endpoints (except DELETE /{id}) have been deprecated from the V1 API, as they only work against Active Directory users. There are new Security/Roles endpoints in the V2 API
Security/Roles(V2) Security/Roles/{id}(V2)	GET, POST, PUT	Added in V2	Security/ Roles API endpoints have been recreated in V2 API to work with both OAUTH and AD users.
Templates/{id}	GET	Changed	Now returns an object with a TemplatePolicy property and a KeyAlgorithms property that show the policies and algorithms the template supports.
Templates/Import	GET, POST	Changed	Now supports multiple algorithms.

Endpoint	Method-s	Action	Notes
Templates/Settings	GET, PUT	Changed	The Template Policy property used to update global application settings now contains four properties: ECDSA, RSA, Ed448, and Ed25519. These replace the AllowEd448, AllowEd25519, RSAValidCurves, and ECCValidCurves.

6.2.1 Incremental Release 11.1 Notes

December 2023

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Important: If you're using the Keyfactor CA Policy Module on one or more certificate authorities, you may have been advised in the past to implement a workaround to resolve an error similar to the following:

```
Could not load file or assembly 'CSS.Common, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=0ed89d330114ab09' or one of its dependencies. The system cannot
find the file specified. at CSS.PKI.X509.X509Utilities..cctor()


2022-07-25 17:06:06.6851 CSS.CMS.CA.Msft.PolicyModule.Policy [Debug] - Certificate
request denied or failed by handler CSSManagedTrustedSubject.PolicyHandler: -
2146233088


denied by policy module
```

The workaround involved creating files *CertSrv.exe.config* and *MMC.exe.config* in C:\Windows\System32 on any certificate authority on which you encountered this issue.

With the release of version 11.0 of Keyfactor Command, this workaround needs to be reversed. If the workaround is not reversed, you will encounter errors such as the following on enrollment attempts:

```
Unable to enroll for certificate. The certificate request failed with the reason
'0x80131902.'


2023-10-24 15:57:07.8547 Keyfactor.CA.Msft.PolicyModule.Policy [Error] - Failure
verifying request: Configuration system failed to initialize


Unrecognized configuration section system.web.
(C:\Windows\system32\certsrv.exe.config line 35)
```

To reverse the workaround, on the affected certificate authority:

1. Rename C:\Windows\System32\CertSrv.exe.config to an alternate name or remove it from the System32 directory.
2. Rename C:\Windows\System32\MMC.exe.config to an alternate name or remove it from the System32 directory.
3. Restart the CA services.
4. Confirm that certificate enrollment is working as desired and that the policy handler(s) in place are working as desired.

New Features

- Agnostic Identity Provider Support for Keyfactor Command.

Updates and Fixes

- Update: Do not display mini table of contents and navigation buttons when the browser is narrower than 880 pixels.
- Update: PFX certificate download includes the option to manually set the password.
- Update: New optional fields for scope and audience are added to the OAuth REST workflow step. When requesting a token during the OAuth REST workflow step, the scope and audience values configured in the step are used if provided.
- Update: New scope and audience configuration parameters can be defined in the OAuth Universal Orchestrator appsettings.json configuration file.
- Update: The colors and fonts on the Management Portal have been changed to match the new Keyfactor branding, including hovers, highlights and buttons changes.
- Fix: Cannot query individual certification in a collection with no global permissions.
- Fix: The TOC button in the documentation did not work in smaller browser sizes.

- Fix: Separate users for appools results in Logi error.
- Fix: Cannot change orchestrator associated with certificate store.
- Fix: Workflow enters/leaves collection timer job causes high SQL load.
- Fix: The IdpHint application setting is removed.
- Fix: A known issue of CA sync using an orchestrator with an invalid CSR causing the sync to stop is fixed. Instead of throwing an exception, such errors will log a warning message, and the record will be skipped.
- Fix: Certificate store reenrollment error when selecting template from separate tenant is fixed. The template dropdown will be filtered to only display the templates associated with the selected certificate authority and based on the CSR Enrollment enabled flag on the templates.
- Fix: The API endpoint GET SSL/Networks reports back an incorrect value for AutoMonitor parameter. The AutoMonitor parameter is removed from the GET SSL/Networks response object.
- Fix: Now the correct notification recipient(s) are shown for field SslAlertRecipients in the response of API endpoint GET SSL/Networks.

Deprecation

- The Classic API was deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11+. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

Known Issues

- .NET Decryption error with separate pool accounts or non-domain joined installation. The solution is to set private key permissions for all app pool users on the encryption certificate.

API Endpoint Change Log

No API endpoint changes were made in this release.

6.3 Major Release 10.0 Notes

September 2022

We're thrilled to announce Keyfactor Command 10.0, which includes some major new features and updates to improve the user experience, enhance automation, and provide native integration with EJBCA.

Important: The MicrosoftECCurveUpgradeModule may fail due to a pre-v10 issue in which Certificate Request Contents were truncated to 4k characters when saved to the database. If your upgrade fails when the MicrosoftECCurveUpgradeModule is run contact Keyfactor Support to obtain assistance with the scripts that will have to be run to fix this issue.

Tip: We encourage customers to contact their customer success manager to discuss the new features and functionality in Keyfactor Command 10, and to schedule an upgrade. Please refer to the *Keyfactor Command Upgrade Overview* for important information about the upgrade process.

Highlights

Workflow Builder

Workflows in Keyfactor Command allow for automation and governance of certificate enrollment and revocation. The workflow builder makes it easy to define workflows within the Keyfactor Command Management Portal to automate event-driven tasks when a certificate is requested (including renewals) or revoked. The workflows can be built with multiple steps between the start and end of the operation that offer a simple way to send notifications, submit approvals, and configure end-to-end automation throughout the environment. This provides for operational agility in an intuitive and easy-to-user tool. Supported built-in steps that can be used in the workflow builder include one or more approval steps supporting one or more approvers, calls to REST APIs, calls to PowerShell, sending emails, and updating enrollment requests with changes to the submitted subject or SANs, if needed. Custom steps can also be built to address specific needs. The workflow builder provides an easy-to-use experience to create rich workflows with multiple steps.

EJBCA Integration with Keyfactor Command

EJBCA is a robust and highly scalable certificate authority. Keyfactor Command now natively integrates with EJBCA version 7.8.1 or higher without the need for a gateway, providing a simpler architecture. The Certificate Authorities area of Keyfactor Command now allows an administrator to enter connection information to an EJBCA CA to manage certificates and support enrollment. With native EJBCA integration, Keyfactor Command offers an alternative to Microsoft CAs. EJBCA is a much more scalable CA with options for multiple CAs on a single server and high availability configuration options that the Microsoft CA lacks. It can also handle a much larger number of certificates than the Microsoft CA.

CA Gateway 22.1 required for Keyfactor Command v10

Upgrade to AnyGateway 22.1 if using gateways on Keyfactor Command v10.

Expanded Template Functionality

- System-wide settings for enrollment templates have moved from the application settings to the templates page.
- Templates can be configured to set policies for the following at both the template level and the system-wide configuration level:
  - Allow Wildcards
  - Allow Public Key Reuse
  - Enforce RFC 2818 Compliance
  - Supported Key Types
- Added a new configuration tab at both the template level and the system-wide configuration level called *Enrollment Defaults* that allows for defining default values for select certificate subject parts that will auto-populate on the PFX Enrollment and CSR Generation pages.
- *Template RegExes* has been renamed to *Enrollment RegExes*. Regular expressions for certificate subject values can be defined at both the template level and the system-wide configuration level.
- Metadata can be configured on a per-template basis to control which fields are shown during enrollment and what default values they have.
- When enrolling with the template, the key size of the request is validated against the template key size. This allows for a key size to be set on a template in Keyfactor Command for validation purposes that can be different than the CA template key size setting.
  - If a CSR Enrollment request is made with a key size that is not valid, per the template policy settings, an error will be displayed when you click the **Enroll** button (for example, the CSR has a key size of 2048 but the template policy supports only 4096).
  - For PFX Enrollment, the request will contain the minimum settings from the Keyfactor Command presiding template settings.
- During the upgrade process Keyfactor Command prevents duplicate template records from being inserted into the database. Duplicate templates could be found if there are templates in different forests with the same name. If you receive an error message during upgrade, contact Keyfactor Support. We will be able to support you through the process of resolving the issue and completing the upgrade. See the *Keyfactor Command Upgrade Overview* for more information.

Keyfactor API Endpoints

The Keyfactor API now has endpoints for most of the functionality found in the product. See the API Endpoint Change Log on page 3105 for information on new and updated API endpoints.

Updates

Changes & Improvements

- **CARecordID Replaces CARequestID**

  The field CARecordID has been added and the field CARequestID has been removed.

- ***Forest* has been Renamed *Configuration Tenant***
  - To broaden Keyfactor Command's compatibility with certificate authorities, the Microsoft-centric term **forest** has been renamed to **configuration tenant**. For EJBCA, there should be one configuration tenant per EJBCA server install. For Microsoft, there should be one per forest. Note that configuration tenants cannot be mixed, so Microsoft and EJBCA CAs cannot exist on the same configuration tenant.
  - Added the ability to search templates by configuration forest and key type. The option to search by forest has been retained for backwards compatibility.

- **SQL Server Connection over SSL**

  As of Keyfactor Command version 10.0, by default Keyfactor Command connects to SQL using an encrypted connection using an SSL certificate configured on your SQL server. Customers should acquire and install an SSL certificate for the SQL server before upgrading to Keyfactor Command 10.0 (see Using SSL to Connect to SQL Server on page 2746 in the *Keyfactor Command Server Installation Guide*). If you would prefer not to use an encrypted channel for your connection to SQL, see Configurable SQL Connection Strings on page 2750.

- **SQL Encryption Key Backup**

  When Keyfactor Command is installed, the option is presented to make a backup of the SQL database master key (DMK). In previous versions of Keyfactor Command, this option backed up the service master key (SMK) instead. For more information about how Keyfactor Command uses the DMK and SMK, see SQL Encryption Key Backup on page 821 in the *Keyfactor Command Reference Guide*.

- **SQL Server 2022 Compatibility**

  Keyfactor Command is compatible with SQL Server 2022.

- **Certificate Requests**
  - The Certificate Requests page is now sorted in descending order by submission date by default. This has been done to cause the more recent requests to appear at the top of the page.
  - The Certificate Requests page is now separated into tabs for pending, external validation, and denied/failed certificate requests.
  - The Denied/Failed tab on the Certificate Requests page now includes only certificate requests denied through Keyfactor Command (see Viewing Certificate Requests on page 163 in the *Keyfactor Command Reference Guide*).
  - The Revoked view filter has been removed from the Certificate Requests page since the expectation is that Keyfactor Command workflows will be used for enrollments and the

history can be viewed as part of that (see Workflow Instances on page 305 in the *Keyfactor Command Reference Guide*).

- **Alerts**
  - When an alert is copied, " - Copy" is appended to the display name to prevent alert display names being duplicated.
  - To aid in clarity, changed the wording on templates when configuring alerts from *None* to *All Templates*.
- **SMTP Application Settings**

  When making changes to the SMTP configuration, the test email can be sent without saving the configuration changes.
- **Certificate Authorities**
  - Added an option to delegate enrollment requests to the Authorization Methods tab. This is in addition to the option to delegate management functions. This allows Keyfactor Command to delegate the authenticated user's credentials to the CA during enrollment to provide end-to-end authentication without unpacking the credentials at the Keyfactor Command layer. If this is not enabled the Restrict Allowed Requesters setting will be used instead. Please see the Authorization Methods Tab on page 367 in the *Keyfactor Command Reference Guide* for more information.
  - When configuring a new certificate authority in the Management Portal, there is now an option to test the connection to the CA before saving the configuration, and CAs will be tested and must be verified and valid to be saved.
  - Updated the CA synchronization so that it logs a message if it could not chain a certificate up to a CA in the system instead of throwing an error.
  - Added a new application setting, *CA Sync Consecutive Error Limit*, which controls the number of times an error can occur before the synchronization job is abandoned.
  - There is no longer the need to register offline CAs, as the root/policy CA certificates can be imported from the issuing CA sync without them. Additionally, the new CA validation makes it impossible to save offline CAs.
- **Certificate Stores**
  - Added the ability for users with only container-level permission to create and use certificate stores in the container, including certificate store types that have a server component. Users will not be able to access certificate stores outside of the containers they have permissions to manage. (Previously, users needed to have Certificate Store Manage permissions in order to change client machine credentials as certificate store servers was shared across all certificate stores with the same type and server name. Now, certificate store servers are partitioned by container.)
  - Added the ability to import PEM certificates that have comments in them when doing an inventory of an F5 REST certificate store.
  - On the Discover tab the label for *Approve* has been changed to *Manage* for clarity.

- **Dashboard and Reporting**
  - The Risk header can now be hidden via security role permissions.
  - Some cosmetic updates have been made to the Risk header.
  - The Collections Dashboard widget is limited to only displaying the first 25 collections configured to be on the dashboard. It sorts the list alphabetically.
  - The stale date is visible in the CRL Monitoring Dashboard widget as a new column and is called *Next Publish by Date*. The stale date should not be used for calculating the status of the CRL. A stale CRL is a valid state and not something that needs to be warned on. If a CRL is stale, the system will check how far it is from expiration and if it is within the warning period it will have a status of *Warning,* or *Valid* if outside the warning period.
  - Keyfactor Command v10 ships with a newer version of Logi Analytics (v14) which drives the Reports and Dashboards. This version provides a number of improvements and fixes some security vulnerabilities.
  - CRL dates are always shown in UTC on the Revocation Monitoring Dashboard.
  - A new report—SSH Key Usage—shows a table which displays a list of SSH keys that have not been used to log on in the given minimum number of days.
  - The Risk header on the dashboard has been updated to avoid awkward text formatting and scrolling when resizing the page.
  - The Risk header titles have been updated for consistency and clarity. Titles referring to expiring certificates are now all in the "Expiring" tense and consistent with each other. *Weak Keys* has been renamed to *Certs with Weak Keys*.
  - The *Certificate Count by Template* report has been updated so that it takes the same parameters as the *Certificate Count per User by Template* report for consistency. This included changing the Evaluation Date to *Start Date* and adding an *End Date* field.
  - All reports have been updated to reference UTC time to avoid confusion about which time zone is being applied.
  - The *PKI Status for Collection* report has been updated to provide clarity on the meaning of *Total Active Certificates*.
- **Agent, Orchestrators, and Orchestrator Management**
  - The Orchestrator Details dialog has been updated to show more information about the orchestrator:
    - Legacy Thumbprint
    - Current Thumbprint
    - Last Thumbprint Used
    - Last Register Status
    - Certificate Rotation Status
  - The Job History now shows the time the job completed.

- The default value for the *Registration Handler Timeout (seconds)* application setting has been extended to 90 seconds for new implementations only. Keyfactor recommends any existing customers using or planning to use custom registration handlers consider extending this timeout to at least 60 seconds.
- SSL scan job parts are now grabbed more deterministically to help keep the job assignments more predictable. For more information, see SSL Network Operations on page 456 in the *Keyfactor Command Reference Guide*.
- The SSL Scan Now option now allows you to select whether to start a discovery job, a monitoring job, or both (see Initiating a Manual Scan on page 466 in the *Keyfactor Command Reference Guide*).
- The Keyfactor Universal Orchestrator now does CRL checking when contacting Keyfactor Command over an encrypted channel (when you configure the orchestrator with a URL referencing https) both when certificate authentication is used and when basic authentication is used. Previously this was only done when certificate authentication was used. If you attempt to connect your orchestrator using SSL and do not have a valid CRL available to the orchestrator, you will get an error message similar to the following:

  ```
  The remote certificate is invalid because of errors in the certificate chain:
  RevocationStatusUnknown, OfflineRevocation
  ```

  For troubleshooting information, see Troubleshooting on page 3003 in the *Keyfactor Orchestrators Installation and Configuration Guide*.

- **Reenrollment**

  A certificate authority and template can now be specified when scheduling a reenrollment job.

- **Certificate Metadata**
  - A certificate metadata field now cannot be deleted if it is in use in a certificate collection definition.
  - When creating a new certificate metadata type, different fields will be displayed depending on the value selected in the Data Type dropdown field. For more information, see Adding or Modifying a Metadata Field on page 710 in the *Keyfactor Command Reference Guide*.
- **Security Identities and Roles**
  - A search bar has been added to search for the collections and containers in the security roles dialog.
  - Improvements were made to performance when loading a large number of security roles in the portal.
  - When copying a security role, a new disclaimer will appear to advise the user that copying a security role will also assign the new role to all the same security identities as the target role.
  - The security roles dialog has been updated to be a tabbed dialog box.
- **UI Changes**

- Some edit dialogs have been changed to use sliding panels to accommodate two different views within the same page rather than pop up windows.
- Added scroll bars to the certificate details pop ups.
- Added the ability to copy data from grid information (e.g. SSL location information when expanding the certificate locations). Information in a grid field can be **copied** to the clipboard by highlighting text in a grid field and clicking **Ctrl**+**C**.
- Performance improvements have been made in loading large data sets in the Management Portal results grids.

- **System Alerts**

  The alerts that are displayed in the UI for notification of things like failed orchestrator jobs have been renamed *System Alerts* for clarity.

- **Logging**
  - The Keyfactor API and Orchestrator API logs on the Keyfactor Command server and the log for the Keyfactor Universal Orchestrator include a correlation ID that helps to identify log messages that originated from the same request. The correlation ID is a randomly generated GUID that often appears just after the date in the log entry and is the same for all log messages for the given request until the request completes.
  - Lowered the logging level for the user's authentication from Info to Trace to avoid cluttering log files.

- **Mac Auto-Enrollment**

  The Mac auto-enrollment process now identifies all the CAs that have the auto-enrollment template(s) available for enrollment and makes a determination as to whether the enrolling user has permissions to enroll on a CA and whether that CA is online before submitting a request to the CA. Previously, a CA was selected randomly among the CAs that had the template(s) available without regard to the user's permissions on the CA or the availability of the CA.

- **Auditing**

  Orchestrator reset, approval, disapproval will now properly audit under the new *Orchestrator* category and their respective operation.

- **Installation**
  - On installation, Keyfactor Command creates an initial record in the DatabaseUpgradeLog table that indicates the exact version of Keyfactor Command that created the database. This can be helpful for troubleshooting.
  - If you are upgrading from an older version of Keyfactor Command the installation directory changed, as of Keyfactor Command v9, to C:\Program Files\Keyfactor. Move any scripts or files that are held in the old directory structure to the new location.

- **Policy Modules**

  The policy modules have been migrated to leverage .NET Core.

- **Custom Registration Handlers**

A custom registration handler can now be designed to enroll against a specific certificate authority and template combination. The registration handler chooses which combination to use. If no combination is requested by the registration handler, then the certificate authority and template from the application settings are used. For more information, see Register a Client Certificate Renewal Extension on page 2961 in the *Keyfactor Orchestrators Installation and Configuration Guide*.

- **Application-Level Encryption Certificate Thumbprint**

  The reference thumbprint for the application-level encryption certificate, if configured, is now stored in the registry on the Keyfactor Command server(s) instead of the SQL database to provide a further level of separation from SQL.

Fixes

- **Keyfactor Command**
  - Revocation Monitoring Dashboard panel no longer stalls as perpetually “Loading” for OCSP endpoints.
  - Certificate subjects for PFX enrollment via the legacy API have been fixed so they can be formatted according to the API.CertEnroll.Pkcs12CertificateSubjectFormat app setting.
  - Fixed an issue when parsing the CSR so that CSRs containing IP or Email SANs no longer cause excess warnings in CA syncs, and IP and Email SANs show up in the pending request details.
  - Fixed an issue where synching external certificates would cause an “object reference not set to an instance of an object” error.
  - Fixed an issue with revocation monitoring alerts reporting time in the local time zone instead of UTC. Emails now have the time in UTC. The time is explicitly labeled UTC.
  - Fixed an issue where special characters like apostrophes would appear HTML-encoded in the collection name.
  - Fixed an issue in certificate enrollment where SANs for IPv4 and IPv6 addresses were not being validated properly.
  - Fixed an issue where an untrusted certificate chain would prevent the certificate details dialog from opening. An error will still occur if a certificate chain is attempted to be downloaded and the chain build fails, but will not prevent the dialog from opening.
  - Fixed an issue where the Identity Audit table wasn’t populating from the Certificate Search page.
  - Fixed an issue where unscheduling an orchestrator management job failed to cancel the previously staged job.
  - Fixed an issue in enrollment where the subject incorrectly added an extra quotation mark when the subject format default was set in certain ways.
  - Fixed an issue where SQL would timeout when deleting over 1,000 certificates from the Keyfactor Command Management Portal.

- Fixed an issue where the gateway configured to run as a domain service account and running on the same server as Keyfactor Command caused RPC errors.
- Fixed an issue where the gateway configured to run as a domain service account caused RPC errors.
- Lowered the logging level for the user's authentication from Info to Trace to avoid cluttering log files.
- Fixed an issue where PEM files with headers could not convert to DER with BouncyCastle 1.9.0 and Keyfactor.PKI.dll v4.x.
- Fixed an issue for certificate store types with the *Advanced>Supports Custom Alias* setting set to **Forbidden**, so that the custom alias should only show on the Add to Certificate Store page when the **Overwrite** checkbox is checked.
- Fixed an issue where using *Delete All* on the Certificate Search page would not delete revoked and expired certificates.
- Fixed an issue in the *Issued Certificates Per Certificate Authority* report that was caused by having templates with the same name in separate forests.
- Fixed an issue with certificate store inventories where a certificate store that had completed an inventory scheduled for an interval would fail if it then was scheduled to run immediately.

- **Keyfactor Agents and Orchestrators**
  - Fixed an issue so that CRLs are now checked regardless of the authentication method being used by the orchestrator.
  - Fixed an issue where permissions were not being set correctly on the appsettings.json and orchestratorsettings.json file that prevented the files being read or updated if the service was running as the Network Service.
  - Fixed an issue where a misconfigured orchestrator using certificate authentication would renew certificate multiple times.
  - Fixed an issue where an orchestrator's registration session was still allowed even when denied by a registration handler and added an auditing event for the orchestrator session registration.

Deprecation

- **Windows Server 2016**

  As of Keyfactor Command version 10.0, Windows Server 2016 is no longer supported.
- **Deprecated Certificate Search Fields**

  The *KeyfactorRequestId*, *RequestResolutionDate*, and *CARequestId* certificate search fields parsers are deprecated due to native EJBCA support in Keyfactor Command as of v10. Any certificate collections using them must be changed before upgrading to v10+.
- **Archive Key on Templates**

As of Keyfactor Command v10 we no longer support enrolling for certificates that have the archive key option turned on in the template to enable the certificate to store the private key for the certificate in the CA. Attempting to enroll using a template that has this option turned on will result in the following error:

```
The certificate request failed with the reason 'The request is missing a required private key for archival by the server.'
```

- **CA Policy module v7.0**

  You will need to upgrade the CA Policy module to v7.1 before running the Keyfactor Command 10.0 upgrade.

- **Reports**

  The Resolution Date field has been removed from the *Certificate Count by User By Template* report.

Future Changes

- **Microsoft .NET Runtime version 3.1**

  By the end of 2022, Microsoft will no longer be supporting .NET Runtime version 3.1. Currently both Microsoft .NET Runtime version 6.0 (x64) and version 3.1 are supported by Keyfactor.

  If you wish to continue using older versions of the Universal Orchestrator but the newer .NET Runtime, you can update the .NET Runtime version on the orchestrator server without needing to reinstall the orchestrator (see System Requirements on page 2880 in the *Keyfactor Orchestrators Installation and Configuration Guide*).

- **Intune Portal/SCEP Change-over**

  Intune portal change-over will be required for SCEP when the old APIs are shut off by Microsoft's deprecation of ADAL at the end of the year.

Known Issues/Limitations

- When editing a template, changes will be lost without warning if the *Save* button isn't clicked before navigating away. This is slated to be fixed in a future release.
- When editing a template, the checkboxes for the Metadata, Enrollment RegExes, and Enrollment Defaults tabs do not allow for multi-edit. This will be fixed in a future release.
- When copying a security role, the identities associated with the security role will also be copied.
- The Condition Variable field in a step of the workflow builder accepts input values that are not valid. Only true, false and variables that will evaluate to true or false are supported.
- For most certificate stores, the *Client Machine* is the machine where the store is located, and the *Orchestrator* drop-down selects the orchestrator/agent. However, for the Java Keystore, the *Client Machine* field is actually the agent and there is no orchestrator dropdown. This will be made more clear in a future release.

- When creating a new certificate store type, the *Depends On Other* option may not be available when creating the parameter. The workaround is to save the certificate store type and then use edit to update the parameter.
- Using the browser back button after generating a report creates a nested instance of Keyfactor Command in Firefox.
- Occasionally, removing a widget from the Dashboard causes the dashboard to hang. Refreshing the browser should resolve this issue.
- The -ne operator in certificate search does not return NULL results for Boolean metadata fields. For a metadata field such as *Unit* use an advanced search such as *Unit -ne "false" OR Unit eq NULL* to get the desired results.
- The *Certificate Count Grouped by Single Metadata Field* report falsely reports no results if using the default metadata value. This will be fixed in a future release.
- The *PKI Status for Collection* report click-throughs do not retain the *Include Unknown* certificates option when clicking through to the certificate search results page. This will be fixed in a future release.
- SMTP Sender information isn't correctly saved by the Configuration Wizard. This will be fixed in a future release. It is recommended to check the SMTP Configuration page upon upgrade.
- Alert tests do not show certificate information if there is no recipient configured to receive an email even if **Send Alerts** is not selected. This will be fixed in a future release. The workaround is to add an email recipient when running the tests.
- Adding multiple enrollment fields at the same time is only saving the last field entered. This will be fixed in a future release. Workaround is to add and save each enrollment field one at a time.
- The *Certificates in Collection* report falsely reports ECC certificates with a certificate state of *Denied* rather than *Active*, revoked certificates with a certificate state of *Active* rather than *Revoked*, and shows a incorrectly shows a revocation reason of *Unspecified* for certificates with an *Active* certificate state. This will be fixed in a future release.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 857: API Change Log

Endpoint	Methods	Action	Notes
/Agents/{id}	GET	Add	
/Agents/Reset	POST	Add	
/AgentBlueprint	GET	Add	
/AgentBlueprint/{id}	GET, DELETE	Add	

Endpoint	Methods	Action	Notes
/AgentBlueprint/{id}/Jobs	GET	Add	
/AgentBlueprint/{id}/Stores	GET	Add	
/AgentBluePrint/ApplyBlueprint	POST	Add	
/AgentBluePrint/GenerateBluePrint	POST	Add	
/Alerts/Denied	GET, PUT, POST	Add	
/Alerts/Denied/{id}	GET, DELETE	Add	
/Alerts/Expiration	GET, PUT, POST	Add	
/Alerts/Expiration/{id}	GET, DELETE	Add	
/Alerts/Expiration/Schedule	GET, PUT	Add	
/Alerts/Expiration/Test	POST	Add	
/Alerts/Expiration/TestAll	POST	Add	
/Alerts/IssuedAlerts	GET, PUT, POST	Add	
/Alerts/IssuedAlerts/{id}	GET, DELETE	Add	
/Alerts/Issued/Schedule	GET, PUT	Add	
/Alerts/KeyRotation	GET, PUT, POST	Add	
/Alerts/KeyRotation/{id}	GET, DELETE	Add	
/Alerts/KeyRotation/Schedule	GET, PUT	Add	
/Alerts/KeyRotation/Test	POST	Add	

Endpoint	Methods	Action	Notes
/Alerts/KeyRotation/TestAll	POST	Add	
/Alerts/Pending	GET, PUT, POST	Add	
/Alerts/Pending/{id}	GET, DELETE	Add	
/Alerts/Pending/Schedule	GET, PUT	Add	
/Alerts/Pending/Test	POST	Add	
/Alerts/Pending/Test/{id}	POST	Add	
/CertificateAuthorities	GET	Update	Schedules are now included in the results.
/CertificateAuthorities	POST	Update	Ability to turn off schedules, sessions are abandoned properly, and threshold monitoring schedule is included.
/CertificateAuthorities/{id}	PUT	Update	Ability to turn off schedules, sessions are abandoned properly, and threshold monitoring schedule is included.
/CertificateAuthorities/{id}	DELETE	Update	Deletion is now prevented if schedules are associated.
/CertificateCollections	POST	Update	Query parameter no longer needed when a valid CopyFromId is provided.
/CertificateCollections/{id}/Permissions	POST	Deprecated	Replaced by /Security/Roles/{id}/Permissions/Collection.
/Certificates/Analyze	POST	Add	
/Certificates/IdentityAudit/{id}	GET	Add	
/CertificateStoreContainers	POST	Add	
/CertificateStoreContainers/{id}	PUT, DELETE	Add	

Endpoint	Methods	Action	Notes
/CertificateStores/Server	GET, POST, PUT	To Be Deprecated	Server usernames, server passwords, and the UseSSL flag are managed by the /CertificateStores API endpoints directly as JobProperties using the Properties parameter, replacing the deprecated /CertificateStores/Server API endpoints.
/CertificateStores	GET, POST, PUT	Updated	Server usernames, server passwords, and the UseSSL flag are managed by the /CertificateStores API endpoints directly as JobProperties using the Properties parameter, replacing the deprecated /CertificateStores/Server API endpoints.
/Enrollment/PFX (v2)	POST	Add	
/Enrollment/Settings/{id}	GET	Add	
/JobTypes/Custom	POST	Update	DefaultValue property is no longer required, validation is now performed on the JobTypeFields/DefaultValue property, validation prevents names containing spaces.
/JobTypes/Custom/{id}	DELETE	Update	Includes validation so that deletion is prevented if at least one associated approved orchestrator implements the capability.
/MacEnrollment	GET, PUT	Add	
/Monitoring/Revocation	GET, POST	Update	Renamed from /Workflow/RevocationMonitoring
/Monitoring/Revocation/{id}	GET, PUT, DELETE	Update	Renamed from /Workflow/RevocationMonitoring/{id}

Endpoint	Methods	Action	Notes
/Monitoring/Revocation/Test	POST	Add	
/Monitoring/Revocation/TestAll	POST	Add	
/Orchestrators/JobHistory	GET	Update	Added JobId field.
/Orchestrators/ScheduledJobs	GET	Add	
/OrchestratorJobs/Reschedule	POST	Add	
/OrchestratorJobs/Unschedule	POST	Add	
/OrchestratorJobs/Acknowledge	POST	Add	
/Security/Identities/{id}	GET	Add	
/Security/Roles/{id}/Identities	GET, POST	Add	
/Security/Roles/{id}/Containers	GET, POST	Add	
/Security/Roles/{id}/Copy	POST	Add	
/Security/Roles/{id}/Permissions	GET	Add	
/Security/Roles/{id}/Permissions/Global	GET, POST, PUT	Add	
/Security/Roles/{id}/Permissions/Collections	GET, POST, PUT	Add	Replaced the /CertificateCollections/{id}/Permissions endpoint functionality.
/Security/Roles/{id}/Permissions/Containers	GET, POST, PUT	Add	Returns only containers that have a permission set for the selected security role.
/SMTP	GET, PUT	Add	
/SMTP/Test	POST	Add	
/Templates	GET, PUT	Update	Includes template-specific policy information.
/Templates/{id}	GET	Update	Includes template defaults.
/Templates/Settings	GET, PUT	Update	Includes global template policies.

Endpoint	Methods	Action	Notes
/Template/SubjectParts	GET	Add	
/Templates/Global/Settings	GET, PUT	Add	
/Templates/Import	POST	Add	
/Workflow/Certificates/Pending	GET	Update	Now supports query fields of Requester and RequestType.
/Workflow/Definitions/Steps/{extensionName}	GET	Add	
/Workflow/Definitions/{definitionId}	GET, PUT, DELETE	Add	
/Workflow/Definitions	GET, POST	Add	
/Workflow/Definitions/Steps	GET	Add	
/Workflow/Definitions/Types	GET	Add	
/Workflow/Definitions/{definitionId}/Steps	PUT	Add	
/Workflow/Definitions/{definitionId}/Publish	POST	Add	
/Workflow/Instances/{instanceId}	GET, DELETE	Add	
/Workflow/Instances	GET	Add	
/Workflow/Instances/My	GET	Add	
/Workflow/Instances/AssignedToMe	GET	Add	
/Workflow/Instances/{instanceId}/Stop	POST	Add	
/Workflow/Instances/{instanceId}/Signals	POST	Add	
/Workflow/Instances/{instanceId}/Restart	POST	Add	

6.3.1 Incremental Release 10.5 Notes

November 2023

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Updates and Fixes

Fix: Certificate authority records could not be saved if they were using PAM for storing credential secrets.

Fix: On a seeded renewal (the Configure option), certificates with multiple SANs did not populate all of the SANs into the renewal form.

Fix: Certificate enrollment regular expressions were not successfully filtering on leading spaces when validating expressions.

Fix: With the application settings for *Allow Custom Friendly Name* and *Require Custom Friendly Name* set to True, on PFX enrollment if the *Include Chain* option was deselected, the value provided in the friendly name field was overwritten with the value from the CN field.

Fix: Certificates originally issued by a Microsoft certificate authority could not be renewed/reissued against an EJBCA certificate authority.

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

Known Issues

- Under some circumstance, the **Test Connection** button on the *Certificate Authority* dialog will erroneously display an error when clicked for a previously saved CA. Despite the error message, the CA still functions (i.e., syncs, enrollments still go through). To work around the error, click the **Save and Test** button.

API Endpoint Change Log

No API endpoint changes were made in this release.

6.3.2 Hot Fix Release 10.4.6 Notes

September 2023

Note: Keyfactor Command 10.4.6 is a hot fix release with a few fixes following the Keyfactor Command 10.4 incremental release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Updates and Fixes

- Update: The data field sizes for the certificate subject field were increased to support re-enrollment jobs for certificates with long subjects.
- Fix: CA synchronizations were hanging with the following error when a corrupted certificate request (for example, a Pending or External Validation record with no CSR) was encountered:

```
Keyfactor.CertificateAuthorityClient.Microsoft.MicrosoftClient [Error] - Value
cannot be null.
Parameter name: inArray
```

- Fix: In an environment with a large numbers of certificate store containers where the orchestrator user did not have global certificate store read permissions, SSL scan times could be excessively long during the certificate import step. This was due to frequent permission check queries on the containers. This hotfix removes unneeded checks. A workaround is to grant the account running the orchestrator the *Certificate Store Management: Read* permission.
- Fix: Workflow instance details for an enrollment request from an enrollment request with multiple SANs of the same type were only displaying the last SAN in the Management Portal. This was

limited to the Management Portal as the Keyfactor API returned the correct data.

- Fix: CSR enrollment with a CSR that included SAN data was also adding any SANs that were provided separately on the CSR enrollment page of the Management Portal, rather than replacing the SANs from the CSR with those entered on the CSR enrollment page. The proper behavior is that the enrollment should use *only* the SANs entered on the CSR enrollment page of the Management Portal if they are provided. If they are not provided on the CSR enrollment page of the Management Portal, the SANs in the CSR will be used on the certificate.
- Fix: SANs entered on the CSR enrollment page of the Management Portal or outside the CSR through the Keyfactor API were not displayed in the workflow instance details page.
- Fix: If the *Allow CSR SAN Entry* application setting was set to false, the Keyfactor API still allowed SANs to be sent in with the certificate request outside of the CSR.

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 858: API Change Log

Endpoint	Methods	Action	Notes
/Enrollment/CSR	POST	Fixed	Includes SANs entered outside the CSR only when the *Allow CSR SAN Entry* application setting is set to true. SANs entered outside the CSR replace SANs in the CSR rather than appending to SANs from the CSR.

Endpoint	Methods	Action	Notes
/Workflow/Instances	GET	Fixed	Includes SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/AssignedToMe	GET	Fixed	Includes SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/My	GET	Fixed	Includes SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/{instanceId}	GET	Fixed	Includes SANs entered outside the CSR in workflow instance details.

6.3.3 Hot Fix Release 10.4.5 Notes

September 2023

Note: Keyfactor Command 10.4.5 is a hot fix release with a few fixes following the Keyfactor Command 10.4 incremental release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Updates and Fixes

- Update: PFX Enrollment, CSR Enrollment, CSR Generation, and certificate renewal all now allow 3072-bit RSA keys.
- Fix: An agent registering with Keyfactor Command was causing SQL locking errors in environments with a large number of scheduled jobs.
- Fix: SANs were not displaying on the pending certificate requests page when a CSR enrollment was done while the *Keyfactor SAN Attribute Policy Handler* was installed on the Microsoft CA and configured for the template that was being used to enroll.

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 859: API Change Log

Endpoint	Methods	Action	Notes
/CSRGerneration/Generate	POST	Update	3072-bit RSA keys are supported.
/Enrollment/CSR	POST	Update	3072-bit RSA keys are supported.
/Enrollment/PFX	POST	Update	3072-bit RSA keys are supported.
/Enrollment/Renew	POST	Update	3072-bit RSA keys are supported.

6.3.4 Hot Fix Release 10.4.4 Notes

August 2023

Note: Keyfactor Command 10.4.4 is a hot fix release with a few fixes following the Keyfactor Command 10.4 incremental release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Updates and Fixes

- Fix: Expiration alert event handlers would fail if they referenced CN or DN in the handler parameters and encountered a null-valued CN or DN certificate during processing of certificates.
- Fix: One-click certificate renewal was not scheduling a job to add the new certificate to the certificate store when the renewal was done by selecting *Continue* instead of *Configure*. The certificate renewal step was completing with both *Continue* and *Configure*.
- Fix: Queries to display a warning about failed or possibly errored orchestrator jobs were impacting SQL performance.

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

API Endpoint Change Log

No API endpoint changes were made in this release.

6.3.5 Hot Fix Release 10.4.3 Notes

July 2023

Note: Keyfactor Command 10.4.3 is a hot fix release with a few fixes following the Keyfactor Command 10.4 incremental release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Updates and Fixes

- Fix: Certificate authorities of EJBCA version 8 could not be added to the certificate authorities page due to a failed version check.
- Fix: One-click renewal was encountering an error when trying to renew against EJBCA version 8.
- Fix: Importing templates to Keyfactor Command from EJBCA version 8 failed.
- Fix: External validation certificates being enrolled from public certificate authorities were sometimes resulting in the following error in the Management Portal with no errors in the log:

  ```
  Cannot convert unidentified or null to object
  ```

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 860: API Change Log

Endpoint	Methods	Action	Notes
/CertificateAuthority/Test	POST	Fixed	EJBCA version 8 is supported.
/Enrollment/Renew	POST	Fixed	EJBCA version 8 is supported.
/Templates/Import	POST	Fixed	EJBCA version 8 is supported.

6.3.6 Hot Fix Release 10.4.2 Notes

June 2023

Note: Keyfactor Command 10.4.2 is a hot fix release with a few fixes following the Keyfactor Command 10.4 incremental release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Updates and Fixes

- Fix: Certificate profiles from EJBCA could not be imported if they were not the first CA listed in the *Allowed CAs* on the certificate profile.
- Fix: End entity profiles configured in EJBCA with *Any CA* in the *Allowed CAs* were not displaying in the Management Portal CSR and PFX enrollment pages.
- Fix: Upgrading to Keyfactor Command version 10 was very slow in environments with a large number of certificate requests in the database.

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

API Endpoint Change Log

No API endpoint changes were made in this release.

6.3.7 Hot Fix Release 10.4.1 Notes

June 2023

Note: Keyfactor Command 10.4.1 is a hot fix release with one update following the Keyfactor Command 10.4 incremental release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://key-factor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Updates and Fixes

- Update: BouncyCastle.Crypto is no longer treated as a shared assembly to allow integrations to be built with newer BouncyCastle classes.

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

API Endpoint Change Log

No API endpoint changes were made in this release.

6.3.8 Incremental Release 10.4 Notes

May 2023

Note: Keyfactor Command 10.4 is a minor release with incremental fixes and updates following the Keyfactor Command 10 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Changes and Improvements

- Workflow now includes two new types—Certificate Entered Collection and Certificate Left Collection—that are designed to help you monitor the comings and goings of certificates from collections and take actions in the event that a certificate unexpectedly appears or disappears from a collection. You might use one of these workflow types to monitor the Weak Keys collection to be alerted via email when a new certificate is added to the collection after being picked up on an SSL scan. Or you might use one of these workflows to monitor a collection of vital certificates and use a PowerShell or REST request to automatically open a support ticket if one of the certificates goes missing. These workflow types work together with the Keyfactor Command Service to periodically evaluate the collections configured for reporting and then initiate workflows for any certificates that have changed membership in the collections. The automated task runs every 10 minutes by default and is not end-user configurable. By default, a maximum of 1000 certificates can be reported on by any one instance of an automated task. This value is end-user configurable with the Concurrent Workflows, setting (see Table 81: Keyfactor Command Jobs Services. Certificate collections that are configured for workflows cannot be edited to prevent triggering a large number of entered/left workflows.
- Certificates with a key type of Dilithium2, Dilithium4, or Dilithium5 may now be imported into Keyfactor Command for management and reporting using the Add Certificate function (see Add Certificate on page 74 in the *Keyfactor Command Reference Guide*). CA synchronization of certificates with this key type will be supported in a future release.
- Certificates with private keys can now be downloaded in JKS format either in PFX enrollment or certificate search. The JKS option for certificates for private keys is in addition to the PEM and PFX options for download format.
- On certificate download in both PFX enrollment and certificate search, you now have the option to select a chain order for the chain certificates in the resulting output file if you opt to include the certificate chain in the download. The choice is either *End Entity First* (at the beginning of the file) or *Root First*.

Updates and Fixes

- Update: The default timeout on the configuration wizard for Keyfactor Command upgrade job executions has been increased to 30 minutes. See Troubleshooting on page 3073 in the *Keyfactor Command Upgrade Overview* for instructions on customizing the timeout.
- Update: The Keyfactor Universal Orchestrator now includes a configuration setting that allows it to skip checking the revocation status (CRL) of the SSL certificate on the Keyfactor Command server when connecting to Keyfactor Command.

- Update: On a new installation of Keyfactor Command, the **Revoke All** option on the Certificates page—controlled with the *Revoke All Enabled* application setting—will default to disabled. This change will not affect existing implementations of Keyfactor Command.
- Update: The wording on the Revoke All option has been changed to clarify that a revocation is occurring.
- Fix: SSL monitoring scans done with the Universal Orchestrator were failing to report TLS 1.3 timeouts.
- Fix: The maintenance job to remove expired stored private keys that are eligible for deletion was not running as expected on a daily basis to remove the keys.
- Fix: A user could be prompted to save changes to a template when viewing a template without making changes in certain template configurations.
- Fix: The certificate template regular expression ^$ to disallow any values in a field was in a catch 22 state requiring entry of a value in the field because a regular expression was defined for it and requiring no value because of the nature of the specific regular expression, causing the field not to function at all.
- Fix: Delegation was not working as expected for certificate revocation when the certificate authority record in Keyfactor Command was configured to *Delegate Management Operations*.
- Fix: Attempting to create records in Keyfactor Command for two certificate stores with the same name on the same server but of different types produced an error indicating that the second was a duplicate of the first; now stores of different types may successfully be created with the same name.
- Fix: Associating a PAM provider with a certificate store container, placing a certificate store in that certificate store container, and then attempting to set the PAM credentials for that certificate store failed with an error of "The supplied Secured Area is invalid for the selected provider".
- Fix: Certificates with a SAN type of *DS Object Guid* could not be imported, producing an error of:

  ```
  illegal object in GetInstance: Org.BouncyCastle.Asn1.DLTaggedObject
  Parameter name: obj
  ```

- Fix: Attempting to validate a CA record for an EJBCA CA using the *Test Connection* option would fail if the client authentication certificate configured for the CA had no EKU defined, resulting in an error similar to:

  ```
  There is a problem validating the CA with ID '3' (check the logs for more details):
  Object reference not set to an instance of an object.
  ```

- Fix: Attempting to disapprove an instance of the Keyfactor Bash Orchestrator when the orchestrator had an SSH synchronization schedule configured or an instance of the Keyfactor Universal Orchestrator when the orchestrator had an SSL scanning schedule configured resulted in a 500 error.

Known Issues

- The Audit Log page offers a search comparison value of *Instance Signal* for the audit category but the results grid Category column references this same value as *Workflow Signal*.
- When a one-click renewal is done on a certificate from the Certificate Search page, even though the renewal succeeds, the grid doesn't refresh with the new status.
- The latest version of the Logi reporting engine has functionality which avoids a system timeout issue by periodically pinging the IIS session behind the scenes so that the dashboard doesn't time out when the session has been idle. As a result, the dashboard no longer refreshes after 20 minutes, but invokes this new functionality instead. The settings used to control this depend on the **Session State Timeout** and **Session Auto Keep Alive** attribute settings in IIS. For more information on this see:

  https://devnet.logianalytics.com/hc/en-us/articles/1500009515942-Manage-Session-Timeout

- On an edit, if you change the workflow step type, you must also change the **Unique Name**. Changing the workflow step type without changing the unique name will result in an error similar to the following:

  ```
  System.Collections.Generic.KeyNotFoundException: The given key was not present in the dictionary
  ```

  Instead of changing both the workflow step type and unique name, you may be prefer to delete the step and create a new step of the desired type.

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in version 11.0 of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 861: API Change Log

Endpoint	Methods	Action	Notes
/Enrollment/CSR	POST	Fixed	Includes SANs entered outside the CSR only when the *Allow CSR SAN Entry* application setting is set to true. SANs entered outside the CSR replace SANs in the CSR rather than appending to SANs from the CSR.
/Workflow/Instances	GET	Fixed	Inclused SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/AssignedToMe	GET	Fixed	Inclused SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/My	GET	Fixed	Inclused SANs entered outside the CSR in workflow instance details.
/Workflow/Instances/{instanceId}	GET	Fixed	Inclused SANs entered outside the CSR in workflow instance details.

6.3.9 Hot Fix Release 10.3.1 Notes

April 2023

Note: Keyfactor Command 10.3.1 is a hot fix release with a few fixes following the Keyfactor Command 10.3 incremental release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Updates and Fixes

- Update: Keyfactor Command disallows installation on Windows Server 2016. This is necessary because Windows Server 2016 does not support PFX files being generated with AES encryption.
- Fix: Expiration reports were failing to generate when the user running the report only had permission to view one certificate collection.
- Fix: Saving a report schedule was failing when the user saving the report only had permission to view one certificate collection.
- Fix: The *Certificate Count Grouped by Single Metadata Field* report was failing when the first metadata field in the report dropdown was left at the default.

Deprecation

- The Classic API will be deprecated in Keyfactor Command version 11.0. All existing uses of the Classic API should be migrated to use Keyfactor API prior to upgrading to Keyfactor Command version 11. If these applications cannot be updated to the newer endpoints then the **Allow Deprecate API Calls** setting must be set to *False* (see Application Settings: API Tab on page 619 in the *Keyfactor Command Reference Guide*). Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.
- The Keyfactor Java Agent will be deprecated in a future version of Keyfactor Command. Customers are encouraged to begin planning a migration to the Keyfactor Universal Orchestrator with the Remote File custom extension publicly available at:

  https://github.com/Keyfactor/remote-file-orchestrator

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 862: API Change Log

Endpoint	Methods	Action	Notes
/Reports/{id}/Schedules	POST	Fixed	Reports can be scheduled when the user scheduling the report only has permission to view one certificate collection.

6.3.10 Incremental Release 10.3 Notes

March 2023

Note: Keyfactor Command 10.3 is a minor release with incremental fixes and updates following the Keyfactor Command 10 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Changes and Improvements

- The new **CertStoreContainer** certificate search field shows certificates in a certificate store that is included in the certificate store container specified by the search criteria.
- The Keyfactor Bash Orchestrator added additional support for using an SSSD user store (e.g. Active Directory) on requests to create logons and distribute key information, allowing keys to be managed for domain users. Domain users can be managed with or without preexisting home directories.
- Added the ability to use any symbols when creating a new SSH logon. This is required in order to facilitate creating a logon for an AD user using SSSD.
- The Universal Orchestrator now communicates with IIS certificate stores over TCP port 445 rather then using WinRM and default ports 5985/5986.
- The BASH Orchestrator now returns improved warning messages on the Job History page. See SSH-Bash Orchestrator Job History Warning Resolution on page 790.

Updates and Fixes

- Update: The Keyfactor Bash Orchestrator now adds the command *restorecon* to the list of commands the orchestrator service account is allowed to execute via sudo on servers running SELinux.
- Update: The Keyfactor Bash Orchestrator now trims Windows line breaks from JSON payloads on send and receive and ignores any data in the authorized_keys file that is not a key (e.g. a comment).
- Update: An application setting—*Enable Legacy Encryption*—has been added to enable/disable the use of legacy encryption methods in PFX enrollment. When the value is set to true, the historical algorithm set (3DES/SHA1/RC2) is used for PFX enrollments. When the value is set to false, the newer algorithm set provided by Windows (AES256/SHA256/AES256) is used instead. The default is *false*.
- Update: A script has been added to allow the Keyfactor CA Policy Module to be upgraded from versions prior to 10.0 and retain existing configuration.

- Fix: EJBCA certificates with a leading zero in the serial number could not be revoked; an attempt to do so generated an error.
- Fix: EJBCA CA Config will give a notification if the certificate you selected doesn't meet requirements, and indicate exactly what the requirements are and what your certificate is lacking.
- Fix: The GET /SSL API endpoint was returning duplicate records.
- Fix: The DELETE /Workflow/Defintions/{id} API endpoint was returning an error if the workflow contained steps.
- Fix: Expiration alert tests displayed a blank dialog if the alert was configured with no recipients.
- Fix: The Keyfactor Bash Orchestrator install failed when the service account was provided an extremely long password.

Known Issues

- The dashboard will throw a secure key error if you let the dashboard sit idle for around 20 minutes. The temporary work-around is to refresh the page. It will be investigated in 11 for a possible fix.
- Because a "+" (plus sign) in a URL can represent either a space or a "+", Keyfactor Command has chosen to read "+" as a space. For CRL URLs that require a "+" (plus sign), rather than a space, replace plus signs in your CRL's URL with "%2B". Only replace the plus signs you don't wish to be treated as a space.
- A user without *Global Certificates - Read* and *Global Certificates – Import* will see a permission error dialog when attempting to view an enrollment workflow instance that has **completed**. The only impact of this error is that it will result in the certificate's information not being parsed in the *Instance Review dialog*. Users should not need these permissions to view their completed workflow instances, and so should not be seeing this error. This will be fixed in the next Keyfactor Command release. The raw data is still present. As a workaround, if a user wants to see the parsed data for that certificate, they would have to use the **KeyfactorId** (found on the workflow instance) in the certificate search page using the **CertId**.

API Endpoint Change Log

No API endpoint changes were made in this release.

6.3.11 Incremental Release 10.2 Notes

January 2023

Note: Keyfactor Command 10.2 is a minor release with incremental fixes and updates following the Keyfactor Command 10 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://key-factor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Changes and Improvements

- **Keyfactor Command (formerly, Timer) Service now runs in an HA environment.**

  The Keyfactor Command (formerly, Timer) service can be installed on every server that Keyfactor Command is installed on. This will allow the service to check out jobs via a locking mechanism which will enforce that any jobs are running on only one service at a time. There is a new *CMSTimerService.exe.config* timeout setting for the service locking mechanism `<add key="Keyfactor.TimerJobs.LockTimeout" value="5000" />` which is the lock timeout. It's the number of ms Keyfactor Command will wait to acquire a lock. By default Keyfactor Command will attempt to get a lock for 5 secs and if unsuccessful, an error will be thrown.

- **Workflow Definitions can be Created via Copy**

  A new option is available on the workflow definitions page that allows you to create a new workflow definition by copying an existing workflow definition. When you create a new workflow definition by copying an existing one, the word "copy" will be appended to the end of the definition name and the workflow key (template) will be cleared. Other data from the copied workflow will be retained.

- **Workflow Step Type Windows Enrollment Gateway - Populate from AD**

  A new workflow step type has been added to support enrollment requests from the Keyfactor Windows Enrollment Gateway using client-side templates configured with the subject as *Build from this Active Directory information*. This workflow step type allows the requests to be completed in Keyfactor Command using an EJBCA template that is not configured to build the subject from Active Directory using the Active Directory information (subject, SANs, and/or SID) supplied in the request from the client.

Updates and Fixes

- Update: The maximum number of characters allowed in a certificate store path has been increased from 256 to 722.
- Update: Users now receive a warning if they attempt to use the Back button in a certificate template after making changes without saving.
- Update: Workflow steps of type Email and Require Approval now go to a failed state if an error occurs in sending an email.
- Fix: An issue encountered with upgrading larger databases in v10.1 is fixed in the current v10.2 release which addressed this specific portion of the database upgrade, and should allow upgrade without this issue.
- Fix: Agent Application Settings: An agent will not attempt to retry a job when this setting is set to `0`.

- Fix: Certificate stores of a type that required a server but did not require authentication to access that server could not be saved using the “No Value” options for the server username and password.
- Fix: A base-64-encoded PEM certificate added to a PEM certificate store using the Certificates -> Add Certificate feature was not being correctly formatted for the store.
- Fix: If multiple template enrollment fields were added at the same time before saving, only the most recently added one was saved.
- Fix: The PKI Status for Collection report drill-downs did not include unknown certificates when the *Include Unknown* box was checked. The *Include Unknown* box also worked inconsistently.
- Fix: Custom orchestrators with a status of Disapproved changed to a status of New when their capabilities were changed. Only orchestrators with a status of Active should change to a status of new when their capabilities are changed.
- Fix: Certificate templates with a key size of Ed448 were imported and assigned a key type of 456.
- Fix: On an attempt to edit the parameters of a built-in report with a parameter of type RelativeDate, an error message appeared indicating “A saved parameter with type ‘RelativeDate’ is invalid with a value of ‘false’” and the user was not allowed to edit the parameters.
- Fix: Chain not being passed in Management Add Job.
- Fix: Certificates cannot be queried by KeyfactorRequestId.

Known Issues

- CSR enrollment fails against a standalone CA. This will be fixed in a future incremental release. Customers using CSR enrollment and standalone CAs should wait to upgrade.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 863: API Change Log

Endpoint	Methods	Action	Notes
/Security/My	GET	Add	Returns all the security roles and global permissions for the requesting user.
/Enrollment/CSR	POST	Update	The workflow instance ID has been added to the response.
/Enrollment/CSR	POST	Update	A new PrivateKey input field has been added to support private key retention on CSR enrollment.

Endpoint	Methods	Action	Notes
/Enrollment/PFX	POST	Update	The workflow instance ID has been added to the response.
/Certificates/Analyze	POST	Update	The endpoint requires Global Certificates-Read or Certificates-Import permissions.

6.3.12 Incremental Release 10.1 Notes

November 2022

Note: Keyfactor Command 10.1 is a minor release with incremental fixes and updates following the Keyfactor Command 10 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 10.0, please review the Major Release 10.0 Notes on page 3094.

Tip: Keyfactor recommends that you check the Keyfactor GitHub Site (https://keyfactor.github.io/integrations-catalog/) with each release that you install to check if you will need to download the updated orchestrators to work with that version of Keyfactor Command.

Changes and Improvements

- **Keyfactor Universal Orchestrator Supports gMSA**

  The Keyfactor Universal Orchestrator now supports running its service as a group managed service account (gMSA).

- **SSL Discovery and Monitoring Jobs have Reset Scan Option**

  A new Reset Scan option has been added for SSL discovery and monitoring jobs that allows to you recover from an SSL job that appears to be stuck or crashed.

Updates and Fixes

- Update: All Keyfactor Command (timer) service jobs have consistent start and stop log messages in both the file and Windows Event Viewer.
- Update: A PAM provider can be used directly by the Keyfactor Universal Orchestrator, such that the server does not retrieve, and does not have access to, the credential.
- Update: Ed448 and Ed25519 keys are now supported for certificate enrollment, policy, import and search.
- Update: Improved support for the Keyfactor Command (timer) service—including a job locking mechanism—in High-Availability implementations.

- Fix: GET /SSL is returning duplicate info in some instances with endpoints sharing a common chain.
- Fix: Certificate store Discovery jobs could not be executed.
- Fix: AnyGateway was declaring all requests as new instead of renew or reissue.
- Fix: The SMTP Sender Account was not populated during the installation and configuration process.
- Fix: SSL discovery scan job errors for entries with a null display name.

Policy Module Updates

- Migrated the Policy Modules to .NET Core 6.
- Updated the Policy Module to create a Windows Event Log entry when the current license is within 60 days of expiration.
- Updated the Policy Module installer to include the EnterpriseLite, SubjectFormat and SCEPRequester modules.
- Updated the Policy Handler Configuration so that changes no longer require the ADCS service to be restarted.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 864: API Change Log

Endpoint	Methods	Action	Notes
/Templates	PUT, GET	Update	Ed448 and Ed25519 keys are now supported for certificate enrollment, policy, import and search.
/Templates/{id}	GET	Update	Ed448 and Ed25519 keys are now supported for certificate enrollment, policy, import and search.
/Templates/Settings	PUT, GET	Update	Ed448 and Ed25519 keys are now supported for certificate enrollment, policy, import and search.

6.4 Major Release 9.0 Notes

August 2021

Release Highlights

We're thrilled to announce Keyfactor Command 9.0, which includes several new features and updates to improve the user experience, deployment flexibility, and risk awareness.

Highlights from the Keyfactor Command 9.0 release are listed here. More details are available in the New Features, and Updates and Improvements sections further down.

> ⚠ **Important:** There have been several UI updates to the navigation menu, drop-downs, and application settings. Thoroughly review these changes in the New Features section.

UI Enhancements

- **What problem does it solve?**

  The Keyfactor Command interface should be easy to navigate and use.

- **How does it work?**

  As we continue to improve the Keyfactor Command interface, we've added updates to the navigation menu, application settings, and dialogues, as well as an updated color scheme.

- **What's the benefit?**

  Ease of Use: The Keyfactor Command interface is more intuitive for new and experienced users alike.

New Risk Header

- **What problem does it solve?**

  PKI administrators and application owners want to easily identify risks and upcoming expirations for the certificates they have access to.

- **How does it work?**

  A new fixed header above the dashboard displays expiring, weak, and revoked certificates for an at-a-glance view of risks.

- **What's the benefit?**

  Risk Mitigation: Enables administrators to quickly identify the state of their certificates.

New Universal Orchestrator

- **What problem does it solve?**

  The current Windows Orchestrator is only able to run on Windows systems.

- **How does it work?**

  The new Keyfactor Universal Orchestrator runs on .NET Core 3.1, which allows it to be installed on servers/instances running either Linux or Windows.

- **What's the benefit?**

Flexibility: Enables customers to deploy orchestrators in cross-platform environments.

New Remote CA Gateway

- **What problem does it solve?**

  Certain customers are unable to use Keyfactor PKI as-a-Service due to security or regulatory requirements, but they'd still like to leverage a SaaS-based solution for certificate management.

- **How does it work?**

  The new Remote CA Gateway securely connects on-premise private PKI – Microsoft ADCS or PrimeKey EJBCA – to the Keyfactor Cloud. This allows customers to leverage Keyfactor Command as a Service (SaaS) while keeping their PKI within their datacenter.

- **What's the benefit?**

  Cloud: On-premise customers now have more options to deploy Keyfactor in a SaaS model – while keeping their PKI in-house, if required.

Support for TLS 1.3

- **What problem does it solve?**

  Before Keyfactor Command 9.0, Keyfactor Command did not support SSL/TLS scanning on endpoints using TLS 1.3.

- **How does it work?**

  The Keyfactor Universal Orchestrator supports SSL/TLS scanning on endpoints using TLS 1.3.

- **What's the benefit?**

  Increased Visibility: Organizations will have improved visibility over certificates.

Template-Level Metadata

- **What problem does it solve?**

  Before Keyfactor Command 9.0, certificate metadata could only be applied system wide.

- **How does it work?**

  Now administrators can apply metadata on a per-template basis, which will override system-wide settings for that specific template.

- **What's the benefit?**

  Control: This gives administrators more granular control for metadata in certificate enrollment.

Ecosystem Updates

While separate from the Keyfactor Command 9.0 release, we've recently introduced several new integrations in GitHub to support more certificate authorities, applications, and services.

These include:

- Google Cloud CA Service: A new AnyCA Gateway implementation supports discovery and automation of certificates issued by Certificate Authority Service (CAS).
- Google Cloud IoT Core: The IoT Issued Alert Handler publishes device certificates to various cloud providers, including Google Cloud, Azure, and AWS.
- GoDaddy: The GoDaddy CA Gateway enables enrollment, renewal, re-issuance, and revocation of certificates via Keyfactor Command.
- Sectigo Certificate Manager: The Sectigo CA Gateway enables full lifecycle management of certificate issued by Sectigo via Keyfactor Command.
- Kubernetes: A proxy signs certificate-signing requests (CSRs) through Keyfactor via the Kubernetes CSR signer API.
- Azure Key Vault: Allows customers to inventory and manage certificates within their Azure Key Vault instances.

More information and developer resources can be found in the Keyfactor GitHub.

New Features

UI Enhancements

Tip: We encourage existing Keyfactor Command customers to watch the Keyfactor Command 9.0 UI Walkthrough demo and read through the detailed UI changes listed below before upgrading to Keyfactor Command 9.

Keyfactor Command 9.0 includes significant updates to the UI, as well as several changes to the main navigation menu and drop-downs with a focus on improved usability. Please continue reading to review and understand these changes.

Previously, the navigation menu looked like the example below:

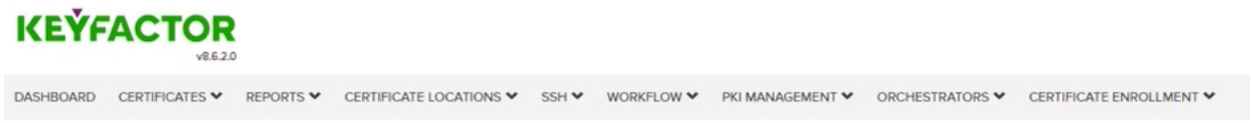


Figure 623: Example Navigation Menu Before Upgrade to 9.0

In Keyfactor Command 9.0, the navigation menu is more concise and user-centric:

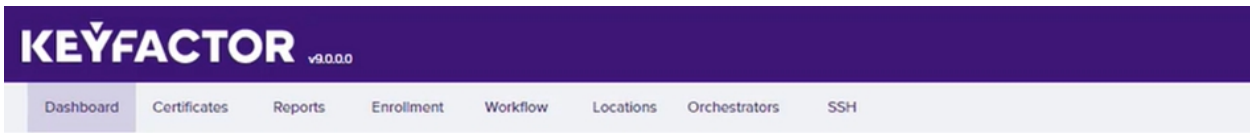


Figure 624: Example Navigation Menu After Upgrade to 9.0

Certificates drop-down

- Add Certificate: The Add Certificate selection is now located in the Certificates tab. Previously, it was accessed via the Certificate Locations tab.

Enrollment drop-down

- Certificate Requests: This option is now found in the new Enrollment tab, rather than the Workflow tab.

Workflow drop-down

- Revocation Monitoring: This option is now located in the Workflow tab. Previously, it was located in the PKI Management tab.
- Expiration: This selection was previously named Expiration Alerts.
- Pending Request: This selection was previously named Pending Request Alerts.
- Issued Request: This selection was previously named Issued Request Alerts.
- Denied Request: This selection was previously named Denied Request Alerts.
- Key Rotation: This selection was previously named Key Rotation Alerts.

Locations drop-down

- Certificate Stores: You will now access the Certificate Stores selection from the new Locations tab. Previously, it was accessed via the Certificate Locations tab.
- Certificate Authorities and Certificate Templates: These menu options are now found in the new Locations. Previously, they were located in the PKI Management tab.
- SSL Discovery: This selection is now located in the Locations tab. It was previously located in the Certificate Locations drop-down.

System Settings menu

- Certificate Store Types: You will now access the Certificate Store Types from the System Settings at the top-right of the screen. It was previously under Certificate Locations.

Certificate Search

- There is a new “ends with” operator. For example:

  ```
  CN -endswith "keyexample.com"
  ```

- A new advanced search option has been added of %ME-AN%. This does a search for account name without domain. For example, the following search in certificate search:

  ```
  NetBIOSRequester -contains "%ME-AN%"
  ```

  Would return certificates requested by the current user as KEYEXAMPLE\jsmith and KEYOTHER\jsmith (assuming the current user is logged in with username jsmith in some domain).

New Risk Header

A *Risk Header* has been added to the Dashboard, which displays relevant information for certificates the user has permissions to. This includes a count of all active certificates, upcoming expirations, expired and revoked certificates, and weak keys (as seen below).

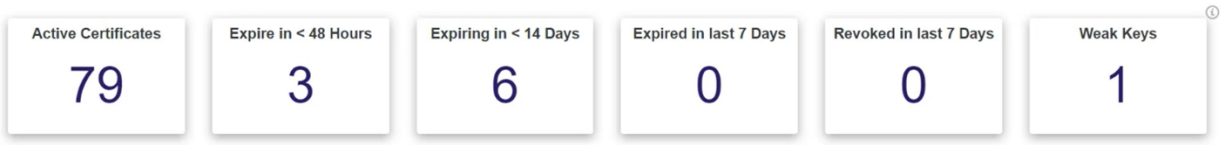


Figure 625: New Risk Header

Note: The new Risk Header is intended to provide an at-a-glance view of key metrics. Unlike items within the dashboard below it, the header cannot be moved or customized.

New Universal Orchestrator

Now available in Keyfactor Command 9.0, the new Keyfactor Universal Orchestrator can perform many of the same functions as the legacy Windows Orchestrator, such as IIS, SSL, FTP and CA management (we will continue to expand its functionality). However, unlike the legacy Windows Orchestrator, the new Keyfactor Universal Orchestrator is able to run on both Windows and Linux servers.

The purpose of orchestrators is to perform SSL scans, manage certificate stores (both Java Key Stores and Windows Certificate Stores), run custom certificate management jobs, inventory CAs, and collect logs to be viewed in the Keyfactor Command Console.

Please review the Deprecation on page 3140 section for more information about the eventual deprecation of the legacy Windows Orchestrator. Refer to the Installing Orchestrators on page 2875 guide for more information on the new Keyfactor Universal Orchestrator.

New Remote CA Gateway

Before Keyfactor Command 9.0, customers had the option to deploy Keyfactor Command on-premise or hosted in the cloud with a fully managed private PKI as a Service (PKIaaS). Now customers have the additional option to keep their PKI on-premise while leveraging Keyfactor Command in the cloud.

The Keyfactor Remote CA Gateway is the connection point between the new Keyfactor Command as-a-Service deployment model (aka Certificate Lifecycle Automation as a Service or CLAaaS) and a customer's on-premise PKI behind their firewall.

The Remote CA Gateway synchronizes in real-time to provide full visibility and governance over the inventory, enrollment, issuance, revocation and renewal of certificates from your on premise CA, requiring just a single, secure API connection on port 443 back to the Keyfactor Command Cloud.

Template-level Metadata

Certificate metadata fields can now be defined on a per-template basis. Before Keyfactor Command 9.0, metadata fields could only be defined as a system-wide setting.

This allows administrators to apply required, hidden or optional settings to a metadata field on a per-template basis so that only certain metadata fields will appear on certain templates.

System-wide settings for metadata fields can be overridden, so customers can choose which fields are displayed, during enrollment for a certificate, based on the template the user selects when enrolling.

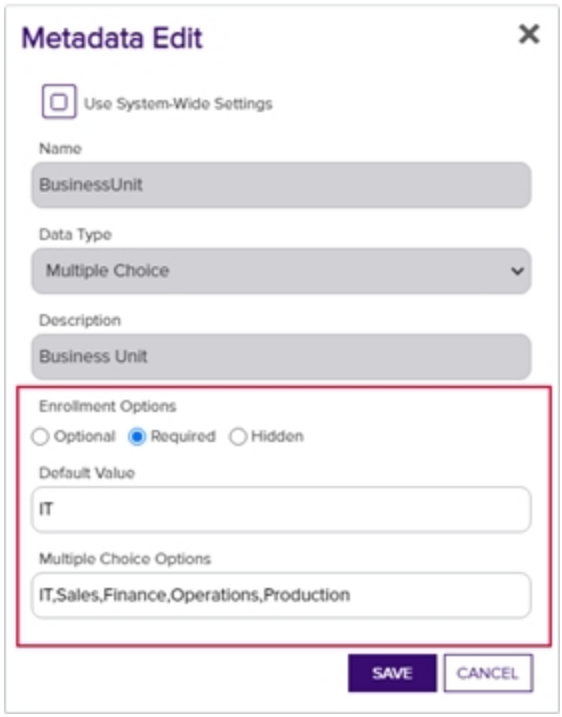


Figure 626: Template Level Metadata

Documentation Structure Updates

Next and Previous buttons have been added to the button row at the top of each page that allow you to navigate through the pages in the documentation in order.

The mini table of contents has been updated to only display by default on pages that contain subpages. This TOC displays—with links—any pages that appear below the current page in the document structure. The TOC button can be used to close and reopen the mini table of contents. The mini table of contents will not display on pages where no subpages are present.

The TOC button now appears when the documents are used in a small browser session (e.g. on a tablet).

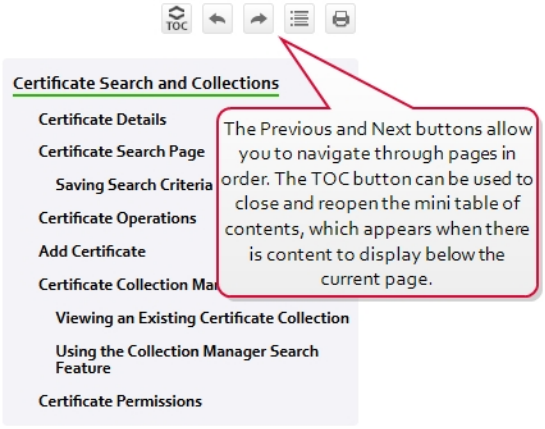


Figure 627: Navigate Forward and Backwards Through Pages

Updates and Improvements

- **Discovery**

  SSL/TLS scanning has been updated to support discovery and monitoring of certificates at endpoints that serve certificates via TLS 1.3. The scan works with the TLS_AES_128_GCM_SHA256 cipher suite. TLS 1.3 connections will also work with SNI.

- **API**

  More API endpoints have been added to do things such as manage security roles, configure certificate store jobs, and manage orchestrators. Please see the Keyfactor API Reference on page 843 for more details. You can access this and the API Endpoint Utility from the portal via the Help icon.

  Additionally, the need for an API application key and secret has been removed. We now control certificate enrollment on the template level within the portal.

- **Logging**

  The log file default locations have moved from C:\CMS\Logs to C:\Keyfactor\Logs. In addition, the NLog.config files have moved from the C:\Program Files\Common Files location to application subfolders of the installation directory, which is C:\Program Files\Keyfactor\Keyfactor Platform by default. Instead of one large CMS_Log file, there are logs for each individual applications.

  See Editing NLog on page 796 in the *Keyfactor Command Reference Guide* for more information.

  Tip: The API is used in conjunction with the applications and both the API log and the relevant other log (e.g. portal) should be consulted when troubleshooting.

- **Administration**

- There is now an option in the Application Settings to require users to agree to Subscriber Terms to enroll for a certificate. This setting also allows administrators to provide a link to those terms.
- CRL Stale Monitoring has been replaced with the ability for customers to define their own definition of "Stale" by generating alerts—and log entries—off the date that the CRL expires, rather than looking at the Next Publish date.

  The main reason for that is that there is, by definition, a race condition between when the new CRL gets created (exactly at the Next Publish time), and when it is copied to the CRL distribution points. Basing alerts off CRL expiration allows customers to tune timeframes based on the way they handle their CRLs.

- **Automation**

  A new constraint has been added to only allow the PowerShell event handlers to run scripts that are located in the path specified in the *Extension Handler Path* in the application settings. By default, this is "C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\". Customers should move scripts to this location or a subdirectory of it and test alerts before going into production. See Adding PowerShell Handlers to Alerts on page 223 in *Keyfactor Command Reference Guide* for more information.

- **Certificates**
  - A new field for *Import Date* has been added to the certificate details page to log when the certificate was imported into the Keyfactor Command database.
  - Certificate Validation now shows the tests that are run when you click on a certificate and the results of those tests.
  - SSL/TLS network name is now displayed on the certificate details dialog.
  - Denied certificate requests now show the denial reason.
  - The CSR generation page has been updated to show the Extended Key Usage of the selected template.

- **Certificates**

  Denied certificate requests are now labeled as *Denied/Failed* to align with public CA terminology.

- **Enrollment**

  Email address subject alternative name option has been added to PFX enrollment.

- **Infrastructure**

  Application pool and service accounts are no longer configured with the db_owner role in SQL, but use a new custom role instead.

- **Orchestrator**

  The certificate thumbprint has been added to the failed job message to help identify which certificate was unable to be deployed to an endpoint.

- **Certificate Authorities**

A new uniqueness constraint has been added to the CertificateAuthorities table. As a result, Keyfactor Command now checks that no CAs share the same logical name and host name combination.

- **Reporting**
  - Added the ability to add a custom logo to scheduled reports.
  - A new report has been added called *Expiration Report by Days* that allows for a number of days to be specified to return a table of the certificates expiring in that timeframe.
  - A column for Reverse DNS has been added to the *Certificates Found at TLS/SSL Endpoints* report.
- **Templates**

  RFC 2818 enforcement has moved from the CA to the template level since different templates have different requirements. Standalone CAs still have the RFC 2818 setting on the CA level.
- **Certificates**
  - Fixed an issue where container level permissions were being ignored during enrollment preventing users from being able to add a certificate to a certificate store in that container.
  - Fixed an issue where regular expressions were being applied to empty values when they should not have been.
- **Dashboard**

  Resolved an issue where the dashboard CRL widget failed to load when configured with a high number of CRLs.
- **Email**

  An issue is fixed where the emails sent from the SSL/TLS scans sometimes reported incorrect totals.

Upgrade Prerequisites

- **Keyfactor Orchestrators**

  We encourage customers to use the new Keyfactor Universal Orchestrator moving forward, which requires .NET Core version 3.1. For existing deployments, .NET version 4.7.2 is required for systems running the legacy Windows Orchestrator.
- **SQL Server 2016**

  Support for SQL Server 2016 has been removed in Keyfactor Command 9.0. Customers should upgrade to SQL Server 2016 Cumulative Update 2 or higher before upgrading to Keyfactor Command 9.0.
- **Database Compatibility**

  Customers will also need to ensure the database compatibility is updated to support 2016 or higher. For more information on updating the compatibility level, please see System Requirements on page 2702 in the *Keyfactor Command Server Installation Guide*.

Upgrade Tasks

Pre-Installation

- If you are using the CA Policy module v7.0 on the same server that the Keyfactor Command Management Portal is installed on, you'll need to upgrade the module to v7.1 before running the Keyfactor Command 9.0 upgrade.
- Upgrade to SQL Server 2016 CU12 or higher and adjust the database compatibility level if needed (see above).

Post-Installation

After the upgrade is complete, some settings will need to be reconfigured due to changes in the way the Keyfactor Command Console handles tasks in Keyfactor Command 9.0:

- RFC 2818 enforcement has moved from the CA to the template level since different templates have different requirements. Standalone CAs still have the RFC 2818 setting on the CA level.
- Configure template-level metadata (if desired).
- Move all Event Handler scripts to the ExtensionLibrary folder under the Keyfactor program installation directory.
- Scripted alert handlers will fail to run if not in the path (or a subdirectory of it) specified by the *Extension Handler Path* application setting. By default, this is "C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\". Customers should move the scripts to this location and test them before moving to production.
- Update any monitoring or other processes that reference the log files to point to the new log file location.

Tip: We encourage customers to contact their customer success manager to discuss the new features and functionality in Keyfactor Command 9, and to schedule an upgrade.

Deprecation

- **API Applications**

  There is no longer the need to configure an API Application in the portal to allow for API enrollment for a certificate with a particular template. Template enrollment permissions are now controlled within the portal on the template level.

- **Classic API**

  The API calls that were previously in the Classic API (CMSAPI) have now been migrated to the Keyfactor API. Customers should use the Keyfactor API going forward and plan to migrate off the CMSAPI in the near future. Support for the CMSAPI will continue for the near future to allow customers time to migrate.

- **Expiration Renewals**

  Existing expiration renewals with Event Handlers will need to have the URLs updated to point to the Keyfactor API instead of the CMSAPI.

- **Windows Orchestrator**

  We will continue to support the Windows Orchestrator. However, all new integrations and extensions will be delivered via the new Keyfactor Universal Orchestrator. We recommend customers use the Keyfactor Universal Orchestrator moving forward as new integrations become available.

- **Verbosity in API Call**s

  In a future version of Keyfactor Command, the API will return all data regardless of the verbosity level. For backwards compatibility where performance is concerned, verbosity will be honored when loading certificate location data in the certificate query but has been replaced with new flags to include this data for future requests.

- **Active Directory**

  In future releases, the ability to use the Active Directory (AD) password on PFX enrollment will be deprecated as we upgrade to allow authentication methods other than AD.

Known Issues/Limitations

Administration

- Daylight Savings Time (DST) is now shown as the time zone locale for clients using Keyfactor Command, rather than the UTC offset, which is the Microsoft CA default. This causes issues during DST to appear off by an hour, in time zones that do not have DST.
- Microsoft IIS settings to change authentication must be made manually to support the *Use Active Directory Password* application setting for the Keyfactor Command Management Portal.
- When using Basic Authentication, the authentication in Microsoft IIS may need to be configured manually for the KeyfactorAnalysis site.
- Authentication between the KeyfactorPortal, KeyfactorAPI, and KeyfactorAnalysis sites needs to be configured with the same authentication type, SSL, and host name.
- On the template RegEx settings, if you unselect use system-wide and do not enter a new RegEx the system-wide RegEx will still apply. To fix this, enter .* in the RegEx field to accept all values.
- When creating a new certificate store type, the *Depends On Other* option may not be available when creating the parameter. The workaround is to save the certificate store type and then use Edit to update the parameter.

Certificates

- Editing certificate details on a collection for a CA, while an initial sync is running on the CA, will cause inaccurate numbers to display in the Edit All window.
- If a CA is not scheduled to sync under Locations, it will not appear in lists to select for things like inclusion in Dashboards and Reports.
- Syncing an Issuing CA before syncing its parents in the chain causes Keyfactor Command to show the wrong requester for the chain certificates.

Keyfactor Command cannot support a CA in the local forest, with the same NetBIOS name as a CA in a trusted forest.

Infrastructure

- Running large SSL scans can impact Keyfactor Command application performance, if the Windows Agent/Orchestrator performing the scan is installed on the same server as the Keyfactor Command portal.
- If you receive an error when opening the portal that “the underlying connection was closed” please be sure you have the latest Windows Updates installed.

Reporting

- In Windows, drive mapping is done on a per-user basis. If you would like scheduled reports to be saved to a mapped drive, the timer service account needs to have that mapping created for them beforehand.
- Exporting a report to Microsoft Excel can fail with a 401 error in Microsoft Edge. Chrome or Firefox can successfully export to Excel. This problem is being worked on by the reporting engine vendor (Logi Analytics).
- Users configured for Logi Analytics reporting cannot have double quotes in the password field.

API

- The GET/Certificates API endpoint has a known issue where if a collection ID is not supplied the request fails. This will be fixed in an incremental release. The workaround in the meantime is to provide a collection ID of zero.

UI

- Occasionally, the “Please Wait” message will hang. Control + F5 will fix this.

Orchestrator

- There is an issue where the Universal Orchestrator is missing a task category in the Windows Event Log and instead reporting a task category of “(16)”. This will be fixed in a future release.
- The new Keyfactor Universal Orchestrator provides much of the same functionality as the legacy Windows Orchestrator (see table below).

Table 865: Keyfactor Universal Orchestrator vs Windows Orchestrator Capabilities

Capabilities	Windows Orchestrator	Universal Orchestrator
IIS Management	✔	✔
CA Synchronization	✔	✔
SSL/TLS Discovery	✔	✔

Capabilities	Windows Orchestrator	Universal Orchestrator
FTP	✔	✔
F5 (SOAP/REST)	✔	
AWS	✔	
NetScaler	✔	
Fetch Logs (new)		✔

New capabilities will be added to the Keyfactor Universal Orchestrator in a future release as we phase out use of the existing Windows Orchestrator over time.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 866: API Change Log

Endpoint	Method	Action	Notes
/Agents/Approve	POST	Add	
/Agents/Disapprove	POST	Add	
/CertificateCollections	PUT	Add	
/CertificateCollections/Copy	POST	Add	
/Certificates/{id}/History	GET	Add	
/Certificates/{id}/Security	GET	Add	
/Certificates/{id}/Validate	GET	Add	
/Certificates/Locations/{id}	GET	Add	
/Certificates/Metadata/Compare	GET	Add	
/Certificates/Metadata/All	PUT	Add	
/Certificates/RevokeAll	POST	Add	

Endpoint	Method	Action	Notes
/CertificateStoreContainers	GET	Add	
/CertificateStoreContainers/{id}	GET	Add	
/CertificateStores/Certificates/Add	POST	Add	
/CertificateStores/Certificates/Remove	POST	Add	
/Enrollment/CSR/Context/My	GET	Add	
/Enrollment/PFX/Context/My	GET	Add	
/JobTypes/Custom	GET, POST, PUT	Add	
/JobTypes/Custom/{id}	GET, DELETE	Add	
/OrchestratorJobs/Custom	POST	Add	
/OrchestratorJobs/JobHistory	GET	Add	
/OrchestratorJobs/JobStatus/Data	GET	Add	
/Reports	GET, PUT	Add	
/Reports/{id}	GET	Add	
/Reports/{id}/Parameters	GET, PUT	Add	
/Reports/{id}/Schedules	GET, POST, PUT	Add	
/Reports/Custom	GET, POST, PUT	Add	
/Reports/Custom/{id}	GET, DELETE	Add	
/Reports/Schedules/{id}	GET, DELETE	Add	
/Security/Identities	GET, POST	Add	
/Security/Identities/{id}	DELETE	Add	
/Security/Identities/Lookup	GET	Add	
/Security/Roles	GET, POST, PUT	Add	
/Security/Roles/{id}	GET, DELETE	Add	
/SSH/Keys/Unmanaged	DELETE	Add	

Endpoint	Method	Action	Notes
/SSH/ServiceAccounts	DELETE	Add	
/SSH/Users/Access	POST	Add	
/SSL/Networks/{id}/Scan	POST	Add	

6.4.1 Incremental Release 9.10 Notes

June 2022

Note: Keyfactor Command 9.10 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

Updates and Improvements

- **Enrollment**

  The enrollment options in Keyfactor Command now support enrolling for SubCA type certificates.
- **Expiration Alert Renewal Handler**
  - Fixed an issue where the expiration alert renewal handler would generate an error if the alert contained more than one email recipient.
  - Fixed an issue where the expiration alert renewal handler would not run on databases that had been upgraded from versions of Keyfactor Command prior to 5.
- **PAM Secret Storage**

  Fixed an issue where PAM parameters of type secret (often passwords) weren't being loaded in Keyfactor Command correctly when returned from the PAM provider.

Note: No changes were made to the Keyfactor Universal Orchestrator in this release, as such the 9.5 version of the Universal Orchestrator should still be considered the latest and is fully compatible with the Keyfactor Command 9.10 release.

API Endpoint Change Log

No API endpoint changes were made in this release.

6.4.2 Incremental Release 9.9 Notes

May 2022

Note: Keyfactor Command 9.9 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

New Features

Metadata Access on View Inventory Dialog

- **What problem does it solve?**

  The View Inventory dialog for certificate stores previously displayed each certificate found in the certificate store but did not include the Keyfactor Command metadata field values configured for the certificates.

- **How does it work?**

  The View Inventory dialog on the Certificate Stores page now includes a Metadata section to allow you to view the metadata fields configured in Keyfactor Command for each certificate found in the certificate store.

- **What's the benefit?**

  Streamlining: You no longer need to look up the metadata fields for the certificates separately.

Updates and Improvements

- **GET /Agents Keyfactor API Endpoint**

  The GET /Agents Keyfactor API endpoint now includes a query parser to allow searching by AgentId. For example:.

  ```
  AgentId -eq "d2f0d545-c3b3-4ea3-bc0a-0232865e24c3"
  ```

- **Logging**

  Changes have been made to the way that Keyfactor Command logs are initialized to support logging from multiple source libraries including Quartz.

- **Alerts Do Not Resume After a Database Connection Failure**

  Fixed an issue in which expiration alerts and pending, issued, and denied certificate alerts that failed due to a database connection problem would not restart on resolution of the database connection issues until the Keyfactor Command service was restarted.

- **Revoke All of Entirely Revoked or Expired Certificates Fails**

Fixed an issue in which attempting to revoke all for a group of certificates that contains only certificates that are revoked already and/or expired results in an error message.

- **SSH Server Groups Incompatible with Domain Names Containing Hyphens**

  Fixed an issue in which SSH server groups could not be created in environments where the Keyfactor Command domain contains a hyphen because the SSH server group owner field would not support a hyphen in the domain name.

- **Certificate Signing Requests Can Produce an Error on Decoding**

  Fixed an issue in which CSR decoder used in CSR enrollment can produce an error on decoding the CSR under select circumstances. These can include SCEP requests with no SANs and CSRs with no extensions.

- **Keyfactor API GET Requests with a Sort Produce a 500 Error**

  Fixed an issue in which Keyfactor API GET endpoints that support query sorting in the URL would produce a 500 error if the sort field was not provided correctly (e.g. the fieldname was entered with a space or was a valid fieldname but not one that was supported for sorting).

Note: No changes were made to the Keyfactor Universal Orchestrator in this release, as such the 9.5 version of the Universal Orchestrator should still be considered the latest and is fully compatible with the Keyfactor Command 9.9 release.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 867: API Change Log

Endpoint	Methods	Action	Notes
/Reports/<any>	GET	Fix	Spaces within the sortField no longer results in an exception.
/Reports/{id}/Schedules	GET	Fix	An invalid sortField no longer results in an exception.
/Agents	GET	Update	New query parser to support the AgentId GUID.

6.4.3 Incremental Release 9.8 Notes

April 2022

Note: Keyfactor Command 9.8 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improve-

ments, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

Updates and Improvements

- **PFX Generation**

  Consolidated PFX generation code so that the PFX files are generated identically from the enrollment and download components.
- **SCEP Intune Integration**

  Keyfactor's Simple Certificate Enrollment Protocol (SCEP) component has been updated to utilize the latest Intune API: Microsoft Authentication Library (MSAL) and Azure AD Graph API.
- **Pending Certificate Request SAN**

  Fixed an issue in which pending certificate requests containing a User Principal Name (UPN) in the Subject Alternative Name (SAN) would be prefixed with '[0]', and IPv6 addresses were not displayed.
- **vSCEP Challenge Error**

  Fixed an issue in which attempting to obtain a Validated SCEP (vSCEP) challenge resulted in an assembly loading error.
- **Denied Alert Email SAN**

  Fixed an issue in which Denied Certificate Alert email did not contain the certificate Subject Alternative Names (SANs).
- **Expiration Alert Logging**

  Fixed an issue in which excessive and superfluous log messages were generated during Expiration Alert processing.

Note: No changes were made to the Keyfactor Universal Orchestrator in this release, as such the 9.5 version of the Universal Orchestrator should still be considered the latest and is fully compatible with the Keyfactor Command 9.8 release.

API Endpoint Change Log

No API endpoint changes were made in this release.

6.4.4 Incremental Release 9.7 Notes

March 2022

Note: Keyfactor Command 9.7 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements,

and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

Updates and Improvements

- **JavaScript Caching**

  Updated pages not to cache static files, including JavaScript.

  Note: After upgrading to 9.7, the cache will still need to be cleared one final time so that the latest version of the pages get loaded with the updated cache setting.

- **API CA Auto-selection**

  The Keyfactor API will auto-select an enrollment certificate authority if one is not explicitly provided.

- **Certificate Stores**

  Fixed an issue in which a user could assign a certificate store to a container without explicit permissions to that certificate store.

- **Certificate Stores**

  Fixed an issue in which database upgrades fail on Azure SQL for newly created databases.

- **Certificate Stores—Scheduling**

  Fixed an issue in which jobs could appear to be scheduled for a certificate store with no available agent.

- **Security Configuration**

  Fixed an issue in which the security roles management page could not be loaded after deletion of an associated Active Directory (AD) group.

- **Metadata String & Integer Fields**

  Corrected an issue where default values could not be set for metadata fields of type string or integer.

- **Certificate Store Deployment**

  Fixed an issue where a certificate cannot be deployed to a certificate store when deploying using a property instead of a certificate store type or Id.

Note: No changes were made to the Keyfactor Universal Orchestrator in this release, as such the 9.5 version of the Universal Orchestrator should still be considered the latest and is fully compatible with the Keyfactor Command 9.7 release.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 868: API Change Log

Endpoint	Methods	Action	Notes
/KeyfactorAPI/License	GET	Add	

6.4.5 Incremental Release 9.6 Notes

February 2022

Note: Keyfactor Command 9.6 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

Updates and Improvements

- **Configuration Wizard**

  Fixed an issue in which the SQL Server login for the application pool account was not created by the configuration wizard.
- **Azure Database Creation**

  Fixed an issue in which database upgrades fail on Azure SQL for newly created databases.
- **Certificate Store—Scheduling**

  Fixed an issue in which jobs rescheduled for *immediate* would not execute.
- **Command Line Configuration Wizard**

  Fixed an issue in which the console configuration wizard cannot populate Azure SQL databases.
- **Custom Orchestrator Job Blueprint**

  Corrected an issue where a duplicate custom job schedule was created when applying the same blueprint to orchestrator.
- **Expiration Report by Days**

  Corrected an issue where the Expiration Report by Days would crash on DD/MM/YYYY formatted dates.
- **Certificate Renewal in Single Store**

  Fixed an issue where a single certificate stored at multiple aliases within the same certificate store was not renewed successfully.

- **CRL Alert Emails**

  Corrected an issue in which a CRL alert email would be sent even if a new CRL was available.

> **Note:** No changes were made to the Keyfactor Universal Orchestrator in this release, as such the 9.5 version of the Universal Orchestrator should still be considered the latest and is fully compatible with the Keyfactor Command 9.6 release.

API Endpoint Change Log

No API endpoint changes were made in this release.

6.4.6 Incremental Release 9.5 Notes

January 2022

> **Note:** Keyfactor Command 9.5 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

Updates and Improvements

- **Agents and Orchestrators**

  Several enhancements have been made to the orchestrators:

  - The alias column size has increased to allow for longer alias names.
  - A new setting allows the IIS stores to be accessed using WinRM over SSL (port 5986).
  - The last thumbprint used for client certificate authentication by orchestrators is now tracked and can be returned using the GET /Agents API method.
  - The UI now allows you to see why an orchestrator could not register for a session rather than having to look in the logs.
  - A new API endpoint has been added to request or require that one or more orchestrators enroll for a new client authentication certificate on the orchestrator's next session registration (POST /Agents/SetAuthCertificateReenrollment).
  - A new API endpoint has been added to reset an orchestrator (POST /Agents/{id}/Reset). Updates include removing orchestrator jobs, deleting associated certificate stores, setting the orchestrator status to new, and clearing thumbprint data as below.
  - The orchestrator reset function in the UI and API now clears the orchestrator client authentication certificate thumbprint data to allow the orchestrator to be reconfigured with a new certificate.

- **Management Portal—Reports**

The “Expiring in less than two weeks” text in the *PKI Status for Collection* report has been updated to change the color scheme to be more readable (white text on a maroon background).

- **API**

  Fixed an issue with the Enrollment/PFX API call not working without specifying a CA. The JobTypes/Custom API call now returns the Job Retry Count.

- **Certificates—Metadata**

  Fixed an issue so that hidden metadata now shows when using *Edit All*.

- **Certificate Stores—Scheduling**

  Fixed an issue to now prompt the user to enter schedule values for *Exactly once* and for *Daily* schedules.

- **Certificate Store—Inventory**

  Fixed an issue when viewing the inventory of certificate store that has an alias without a certificate.

- **Installation—Modify/Remove**

  Corrected an issue where the MSI would freeze if trying to modify or uninstall an installation that had been done without any components selected to be installed.

- **Orchestrators and Agents—Custom Job Retry**

  Corrected an issue where custom jobs would not retry if the job complete handlers failed.

- **Alerting—Email Address Format**

  Fixed an issue where the email address validation was not allowing some valid subdomains.

- **Registration Handler—Enrollment**

  The registration handler now receives the certificate chain for enrollments performed via the enrollment callback.

- **Management Portal Reports**

  Updates to Management Portal reports to handle upgrade scenarios and other user interface fixes.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 869: API Change Log

Endpoint	Methods	Action	Notes
/Enrollment/PFX	POST	Update	No longer requires a certificate authority name to be provided.

6.4.7 Incremental Release 9.4 Notes

December 2021

Note: Keyfactor Command 9.4 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

Updates and Improvements

- Log4j CVE Vulnerability

  Keyfactor has conducted an assessment of the recently-announced CVE for the log4j library (https://github.com/advisories/GHSA-jfh8-c2jp-5v3q). We have identified that the vast majority of the Keyfactor suite of products are NOT affected. This includes EJBCA, SignServer, the Keyfactor Command platform, Keyfactor Control, and Code Assure.

  The only component that does make use of the log4j library is the Java Agent for Keyfactor Command; for clarity, all other Keyfactor agents and gateways are NOT affected.

  Details

  According to the CVE, exploit of the vulnerability requires compelling log4j to log user-controlled input. In the case of the Java agent, there are mitigating factors, such as:

  - The Java agent has an “outbound-only” connection pattern and does not accept inbound network connections of any kind.
  - Users of the Java agent who could control such input are typically Keyfactor administrators.
  - The limited nature of things the Java agent is expected to log.

  From Log4j – Apache Log4j Security Vulnerabilities:

  - Mitigation: This behavior can be mitigated by setting either the system property log4j2.formatMsgNoLookups or the environment variable LOG4J_FORMAT_MSG_NO_LOOKUPS to true.

  Patch Implementation—The 8.7.2 version of the Java Agent to utilize the patched version of Log4j, and mitigate the vulnerability.

- **Orchestrator Certs**

  Ability for an orchestrator to use a TLS client authentication certificate to map to a Windows identity in IIS and to use a different TLS certificate provided in an HTTP header to identify the orchestrator to Keyfactor Command.

- **External Validation Certificate Requests**

  Certificate requests returning a status of EXTERNAL_VALIDATION are not treated as failures and will be sync’d with appropriate metadata when the certificate is available.

- **Certificate Detail Data Efficiency**

  The certificate details are obtained from the server when needed, and not as part of the initial certificate query. This greatly increases the efficiency and performance of the page.

- **Query Optimization for Large Scale Environments**

  Multiple optimizations have been made to improve management portal query performance, scalability, and stability in large scale environments.

- **Pending Certificates API Endpoint**

  Metadata for certificate requests in a pending state is now available for retrieval via the /Workflow/Certificates/Pending API endpoints (GET /Workflow/Certificates/{id} and GET /Workflow/Certificates/Pending).

- **SSL Scanning Chunk Sizes**

  Distinct SSL scanning chunk size application settings are now available for discovery and monitoring to allow for greater control over performance tuning.

- **Dashboard Risk Header Clarifications**

  The dashboard Risk Header now contains verbiage to clarify that no filtering exists for renewed certificates in expired query counts.

  In addition, the dashboard Risk Header contains verbiage noting that the certificate counts are global and not limited to only those to which the current user has access.

- **Custom Job Blueprint Duplication**

  An issue was fixed so that a copy operation on a blueprint successfully copies custom jobs.

- **Certificate Count by Template Report**

  An issue was fixed so to properly retain the selected default certificate authority.

- **SSL Quiet Hours Daylight Savings**

  Updates were made to the SSL Quiet Hours to better handle schedules involving Daylight Savings Times.

- **SSL Monitoring Emails**

  SSL Monitoring emails now send the complete and correct data when multiple orchestrators are in simultaneous use.

- **Certificate Detail Before/Not After Dates**

  Certificate details now display the time in addition to the date for Before and Not After dates.

- **SSL Scanning Certificate History**

  A fix was implemented to properly display the history of certificates imported into the system via SSL scanning.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 870: API Change Log

Endpoint	Methods	Action	Notes
/Workflow/Certificates/Pending	GET	Update	Now returns the associated metadata.

6.4.8 Incremental Release 9.3 Notes

November 2021

Note: Keyfactor Command 9.3 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

Updates and Improvements

- **Certificate Search**

  The certificate search functionality has been optimized to increase speed and efficiency, especially with higher numbers of certificates and associated certificate locations. This means certificate searches done in the management portal for large data sets that include certificates found in certificate stores (e.g. 250,000+ certificates each in 5 or more certificate stores) now complete more quickly.
- **Failed Certificate Management Jobs**

  Certificate management jobs that have failed no longer continue to run.
- **PKI Status Report Time Zone**

  Corrected the format of time zones in the PKI Status for Collection Report.
- **Database Encryption Configuration**

  The Configuration Wizard now verifies the selected database encryption certificate has an associated valid private key.
- **SSL Scanning**

  Updates made to the SSL scanning process to be more efficient and eliminate potential process-locking scenarios.
- **Management Portal User Interface**

  Various Management Portal user interface fixes.
- **Management Portal Reports**

  Updates to Management Portal reports to handle upgrade scenarios and other user interface fixes.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 871: API Change Log

Endpoint	Methods	Action	Notes
/JobTypes/Custom	POST	Fix	No longer requires default field values.

6.4.9 Incremental Release 9.2 Notes

October 2021

Note: Keyfactor Command 9.2 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

New Features

UI Support for PAM CA Password Entry

- **What problem does it solve?**

  The API previously supported the entry of certificate authority passwords to be stored within a Privileged Access Management (PAM) instance, but the UI did not implement this functionality.

- **How does it work?**

  The certificate authority editor dialog allows for entry of a password to be stored in a PAM instance.

- **What's the benefit?**

  Flexibility: Allows for multiple ways to securely store and manage certificate authority passwords

Custom Orchestrator Bulk Scheduling

- **What problem does it solve?**

  Custom orchestrator jobs can currently only be scheduled individually.

- **How does it work?**

An API endpoint (POST OrchestratorJobs/Custom/Bulk) has been created to implement bulk schedules. The job identifiers along with the desired schedule can be provided in a single call.

- **What's the benefit?**

  Ease-of-Use: Enables administrators to easily schedule large batches of custom orchestrator jobs.

Updates and Improvements

- **CA Management with PAM**

  When configuring the *Use Explicit Credentials* option on a CA, you can now choose a PAM provider as the storage location for the credential password or the Keyfactor secrets table.

- **Logi Analytics License**

  A new license for Logi Analytics is required as the previous version is expiring. The 9.2 release includes the license update. Please see Updating Logi Analytics License on the next page for more information.

- **CSR Parsing Containing Spaces**

  CSRs containing spaces can now be parsed successfully during enrollment.

- **Robust SSL Certificate Parsing Error Handling**

  Certificates that fail to be parsed during SSL scanning are now logged but do not cause the entire scan to immediately fail.

- **Robust Alert Failure Error Handling**

  A failure processing an alert no longer prevents processing of subsequent alerts.

- **Hidden Metadata Enrollment Fields**

  Metadata fields which are hidden during the enrollment process are now displayed properly in the resulting certificate details.

- **Collection-based Reports Failing**

  Reports based on collections containing Revocation, Certificate State or Common Name no longer fail.

- **Incorrect CSR Enrollment CA**

  The proper forest certificate authority is used for enrollment when using the API to enroll via CSR.

- **Denied Alerts Template**

  The Denied Certificate Request alerts are once again properly scoped to the selected template. This was a regression from a previous release.

- **Java & C Agent Inventory Error**

  An error was corrected in which an error was thrown if no entry updates were returned during inventory processing.

- **Orchestrator/Agent Re-Enrollment Error**

Fixed an issue in which an object reference error was thrown during re-enrollment operations.

- **Orchestrator Ceases Processing after Batch Submission**

  Corrected an issue in which the orchestrators would cease processing after submission of a large batch of SSL results.

Updating Logi Analytics License

Logi is a 3rd party BI tool which is used by Keyfactor Command for its dashboard and report features. The license required for Logi is integrated into Keyfactor Command and resides within the product's Logi folder. The license's current term is 3 years with a 7-day grace period after expiration. During that grace period, an alert will appear, and a new license should be used to remediate the issue. Here are two examples:

- License close to expiration:

  Figure 628: Keyfactor Logi License Expiration Alert

  Dashboard:

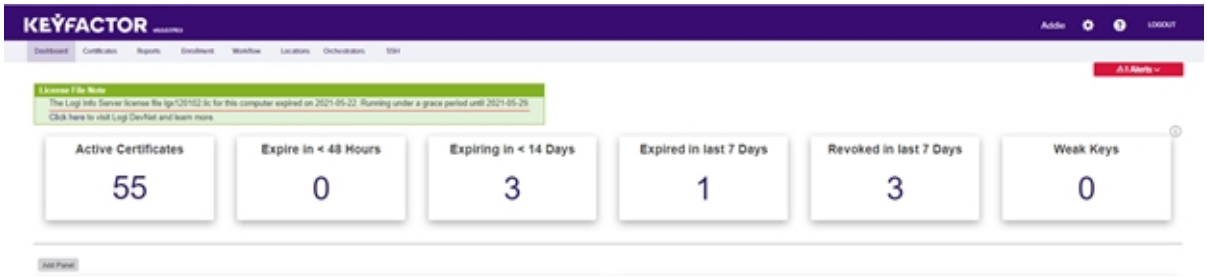
  

  Figure 629: Keyfactor Logi License Expiration Alert on the Dashboard

  Report:

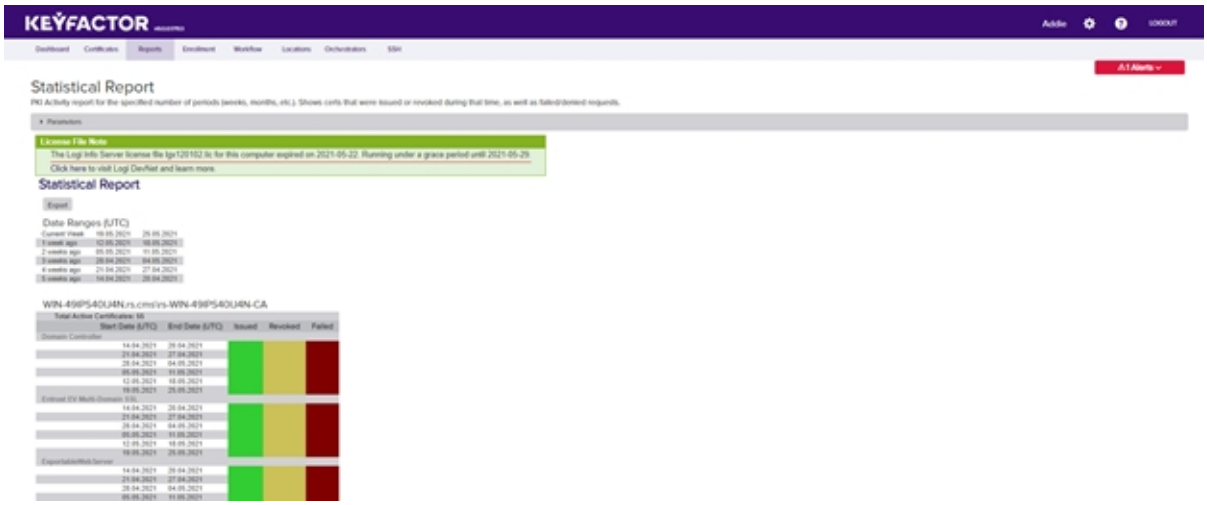


Figure 630: Keyfactor Logi License Expiration Alert on Report

- Expired license:

  The Dashboard and Reporting capability is not available with an error message displayed like the one below.

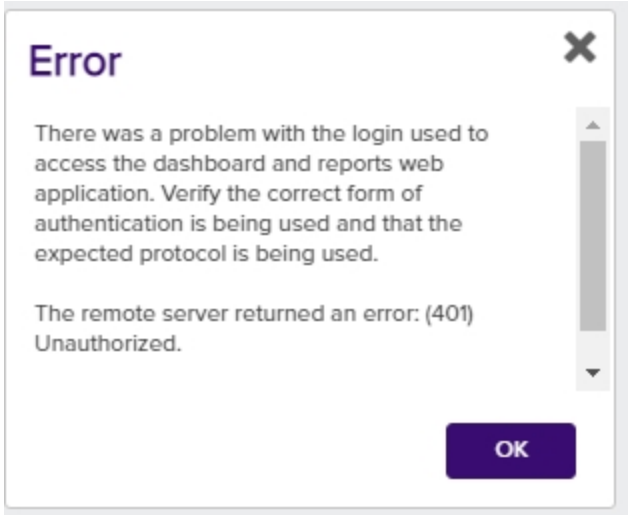


Figure 631: Keyfactor Expired Logi Error Message

Solution

The updated license for Logi is included in release 9.2 and will be installed automatically as part of the upgrade to or fresh installation of this version. If you are not installing Keyfactor Command v9.2, replace the license manually as follows:

1. On your Keyfactor Command server, navigate to the Logi folder in your Keyfactor Command instance. By default, this is:

   C:\Program Files\Keyfactor\KeyfactorPlatform\Logi

If you are on an earlier version of Keyfactor Command your license file will by default be found in the following directory:

C:\Program Files\Certified Security Solutions\Certificate Management System\Logi]

2. The license file ends with an extension of *.lic*. Replace the license file with a valid one provided to you by Keyfactor. The license filename cannot be changed and should remain as "lgx120102.lic".

If the license has already expired, once it is replaced with a valid one and the browser is refreshed, the product will work as expected. The alert will no longer appear.

If you upgrade to a version of Keyfactor Command prior to v9.2 after replacing the license file, you will need to manually add the new license file again.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 872: API Change Log

Endpoint	Methods	Action	Notes
/Certificates	GET	Fix	No longer fails if a collection id is not provided.
/OrchestratorJobs/JobHistory	GET	Fix	Request no longer fails for 'Dynamic' job types.
/Reports/Schedules/{id}	DELETE	Fix	Response code is now 200 when the user role does not have *Modify – Report* permission.

6.4.10 Incremental Release 9.1 Notes

September 2021

Note: Keyfactor Command 9.1 is a minor release with incremental fixes and updates following the Keyfactor Command 9 major release. For more details on new features, improvements, and known limitations in Keyfactor Command 9.0, please review the Major Release 9.0 Notes on page 3130.

New Features

Custom Certificate Store Parameters

- **What problem does it solve?**

  Provide the ability to associate custom parameters with certificate stores and certificate store entries to associate useful information.

- **How does it work?**

  The certificate store type dialog now provides tabs for entry parameters, in addition to custom fields. These parameters and custom fields can be defined for input during enrollment, storage and management of certificate store inventory. For more information, see Entry Parameters Tab on page 707 in the *Keyfactor Command Reference Guide*.

- **What's the benefit?**

  Flexibility: Allows for further customization around certificate stores which can be dictated by customizable data.

Certificate Store Inventory

- **What problem does it solve?**

  The previous version of certificate store inventory leveraged the certificate search functionality. While this worked, it was not always well-suited for the viewing of certificate store inventory.

- **How does it work?**

  Clicking on the *View Inventory* button with a certificate store selected will now load a dialog with the inventory of the store.

- **What's the benefit?**

  Ease-of-Use: Enables administrators to efficiently review certificate store inventory.

Certificate Store Type Parameters

- **What problem does it solve?**

  The previous certificate store type parameters were defined via a comma-separated list and were not strongly typed.

- **How does it work?**

  A formalized list is available to define parameters explicitly, including type (String, Boolean, Multiple Choice, Secret).

- **What's the benefit?**

  Flexibility: Enables more powerful definition of certificate stores and data-validity checking.

Certificate Store Parameter Reporting

- **What problem does it solve?**

  The current on-boarding of certificate stores requires manual data entry of custom fields and parameters.

- **How does it work?**

  The Keyfactor Command orchestrator framework provides for orchestrators to report certificate store entry parameters.

- **What's the benefit?**

  Flexibility: Enables customers to more easily track new certificate stores and changes to them made out-of-band from Keyfactor Command.

Keyfactor Command Configuration Wizard

The Keyfactor Command server configuration wizard now supports entry of group managed service accounts (gMSA) in the Administrative Users field on the Keyfactor Portal tab.

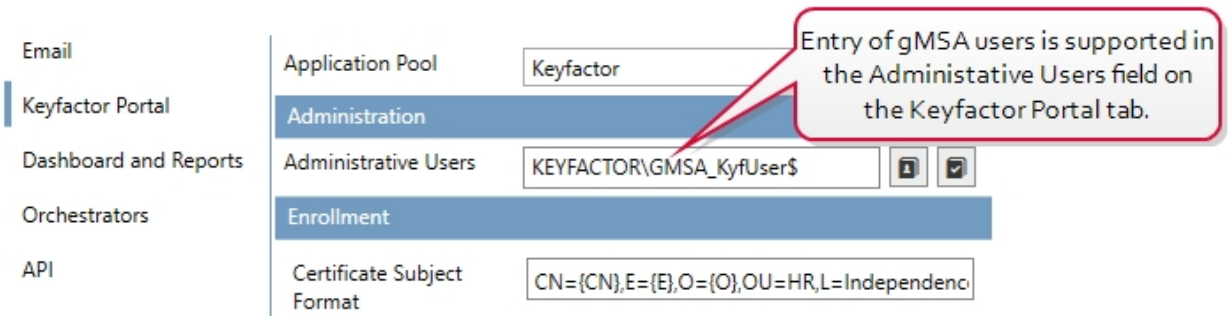


Figure 632: Entry of gMSA Users in the Administrative Users Field

Note: Entry of gMSA users is not supported in the fields that require entry of a password in the configuration wizard (e.g. the service account on the Service tab) at this time. GMSA users cannot be selected using the people picker.

Updates and Improvements

- **Job Completion**

  Job completion handler is now provided the certificate identifier upon renewal so that the handler can perform any related tasks.

- **API Endpoint Deprecation**

  The CertificateCollections/{id}/Permissions endpoint due to an update slated for the Keyfactor Command v10 release and the fact that the endpoint is not updating permissions properly.

- **Permissions Message**

An incorrect error message was displayed to users without sufficient permissions to a certificate collection.

- **Certificate Store Deletion**

  Fixed an issue in which a Certificate Store cannot be deleted if there is a job staged against it.

- **Pending Alerts**

  Pending alerts were being sent on certificate issuance regardless of the associated template.

- **Certificate Inventory**

  Corrected a permissions problem in which users with only read permissions on a Certificate Store were unable to view inventory of that certificate store.

Known Issues

- **CSR Enrollment**
  In cases where there are duplicate template names in multiple forests, CSR enrollment can sometimes go to the wrong CA. This will be fixed in a future incremental release. Customers with environments with duplicate templates should wait to upgrade.

API Endpoint Change Log

The following changes were made to the API endpoints. Please review these carefully if you have implemented any integration using these endpoints.

Table 873: API Change Log

Endpoint	Methods	Action	Notes
/CertificateStores/{id}/Inventory	GET	Add	
/Enrollment/PFX/Replace	POST	Fix	SuccessfulStores collection now only includes Ids of stores that were successfully processed.
/Enrollment/PFX/Deploy	POST	Update	Now allows for multiple stores of the same type with different parameters.
/CertStoreTypes	POST/PUT	Update	EntryParameters can now be set via these methods.
/CertificateStores/Certificates/Add	POST	Update	Now allows for multiple

Endpoint	Methods	Action	Notes
			stores of the same type with different parameters.
/CertificateStores/Certificates/Remove	POST	Update	Now allows for multiple stores of the same type with different parameters.
/CertificateCollections/{id}/Permissions	GET	Deprecate	

6.5 Keyfactor Command v11 Compatibility Matrix

Keyfactor Command version 11's compatibility with the various supported Keyfactor gateways, agents and orchestrators is shown in the table below (see also Keyfactor Command Compatibility Matrix Legend 1).

Table 874: Compatibility Matrix for Keyfactor Command v11

Product	Keyfactor Command Compatibility
Universal Orchestrator	

Version	11.0
11.0.0	✓
10.4.1	
10.4.0	
10.2.0	
10.0.1	
9.4.0	
9.2.0	
9.0.2	

Product	Keyfactor Command Compatibility
Windows Orchestrator	Version: 8.7.2 — 11.0: ✗
EJBCA Native Support	Version: 8.0.0.0 — 11.0: ✓ 7.12.0.1 — ✗ 7.12.0.0 — ✗ 7.11.0.0 — ✗ 7.10.1.0 — ✗ 7.10.0.1 — ✗ 7.9.1.0 — ✓
SSH	Version: 2.0.0 — 11.0: ✓ 1.0.1 — ■
Java Agent	Version: 8.7.2 — 11.0: ✓
AnyCAGateway REST	Version: 23.1 — 11.0: ✓

Product	Keyfactor Command Compatibility
AnyGateway	20.9 / 11.0 23.3.0 ✳ ✓ 22.1.1 ✳ ✓ 22.1.0 ✳ ✓ 21.10.x ✳ ✗ 21.5.x ✳ ✗ 21.3.x ✳ ✗ 20.9 ✳ ✗ 20.7 ✳ ✗
Windows Enrollment Gateway	Version / 11.0 23.3 ✓ 23.1.0 ✓
SQL Server	Version / 11.0 2016 ✗ 2017 ✓ 2019 ✓ 2022 ✓
Windows Server	Version / 11.0 2016 ✗

Product	Keyfactor Command Compatibility

Version	11.0
2019	✓
2022	✓

6.6 Keyfactor Command v10 Compatibility Matrix

Keyfactor Command version 10's compatibility with the various supported Keyfactor gateways, agents and orchestrators is shown in the table below (with Keyfactor Command Compatibility Matrix Legend 1).

Table 875: Compatibility Matrix for Keyfactor Command v10

Product	Keyfactor Command Compatibility

Universal Orchestrator

Version	10.4	10.3	10.2	10.1	10.0
10.4.1	✓	✓	✓	✓	✓
10.4.0	✓	✓	✓	✓	✓
10.2.0	✓	✓	✓	✓	✓
10.0.1	✓	✓	✓	✓	✓
9.4.0	✓	✓	✓	✓	✓
9.2.0	✓	✓	✓	✓	✓
9.0.2	✓	✓	✓	✓	✓

Windows Orchestrator

Version	10.4	10.3	10.2	10.1	10.0
8.7.2	✓	✓	✓	✓	✓

Product	Keyfactor Command Compatibility

EJBCA Native Support

Version	10.4	10.3	10.2	10.1	10.0
8.0.0.0	✗	✗	✗	✗	✗
7.12.0.1	✓	✓	✓	✓	✓
7.12.0.0	✓	✓	✓	✓	✓
7.11.0.0	✓	✓	✓	✓	✓
7.10.1.0	✓	✓	✓	✓	✓
7.10.0.1	✗	✗	✗	✗	✗
7.9.1.0	✓	✓	✓	✓	✓

SSH

Version	10.4	10.3	10.2	10.1	10.0
2.0.0	✓	✓	■	■	■
1.0.1	■	■	✓	✓	✓

Java Agent

Version	10.4	10.3	10.2	10.1	10.0
8.7.2	✓	✓	✓	✓	✓

AnyCAGateway REST

Version	10.4	10.3	10.2	10.1	10.0
23.1	✗	✗	✗	✗	✗

AnyGateway

20.9	10.4	10.3	10.2	10.1	10.0
23.3.0 ✳	✓	✓	✓	✓	✓
22.1.1 ✳	✓	✓	✓	✓	✓
22.1.0 ✳	✓	✓	✓	✓	✓

Product	Keyfactor Command Compatibility

20.9	10.4	10.3	10.2	10.1	10.0
21.10.x ✳	✗	✗	✗	✗	✗
21.5.x ✳	✗	✗	✗	✗	✗
21.3.x ✳	✗	✗	✗	✗	✗
20.9 ✳	✗	✗	✗	✗	✗
20.7 ✳	✗	✗	✗	✗	✗

Windows Enrollment Gateway

Version	10.4	10.3	10.2	10.1	10.0
23.3	✓	✓	✓	✗	✗
23.1.0	✓	✓	✓	✗	✗

SQL Server

Version	10.4	10.3	10.2	10.1	10.0
2016	✗	✗	✗	✗	✗
2017	✓	✓	✓	✓	✓
2019	✓	✓	✓	✓	✓
2022	✓	✓	✓	✓	✓

Windows Server

Version	10.4	10.3	10.2	10.1	10.0
2016	✗	✗	■	■	■
2019	✓	✓	✓	✓	✓
2022	✓	✓	✓	✓	✓

6.7 Keyfactor Command v9 Compatibility Matrix

Keyfactor Command version 9's compatibility with the various supported Keyfactor gateways, agents and orchestrators is shown in the table below (with Keyfactor Command Compatibility Matrix Legend 1).

Table 876: Compatibility Matrix for Keyfactor Command v9

Product	Keyfactor Command Compatibility									
Universal Orchestrator	**Version**	**9.10**	**9.9**	**9.8**	**9.7**	**9.6**	**9.5**	**9.4**	**9.3**	**9.2**	**9.1-9.0**
	10.4.1	■	■	■	■	■	■	■	■	■	■
	10.4.0	■	■	■	■	■	■	■	■	■	■
	10.2.0	■	■	■	■	■	■	■	■	■	■
	10.0.1	■	■	■	■	■	■	■	■	■	■
	9.4.0	✓	✓	✓	✓	✓	✓	✓	■	■	■
	9.3.0	✓	✓	✓	✓	✓	✓	✓	✓	■	■
	9.2.0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	9.0.2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Product	Keyfactor Command Compatibility

Windows Orchestrator

Version	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
8.7.2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

EJBCA Native Support

Version	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
8.0.0.0	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
7.12.0.1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
7.12.0.0	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
7.11.0.0	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
7.10.1.0	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
7.10.0.1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
7.9.1.0	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Product	Keyfactor Command Compatibility

SSH

Version	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
2.0.0	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
1.0.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Java Agent

Version	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
8.7.2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

AnyCAGateway REST

Version	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
23.1	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

AnyGateway

20.9	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
23.3.0 ✳	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
22.1.1 ✳	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨

Product	Keyfactor Command Compatibility

20.9	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
22.1.0 ✳	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
21.10.x ✳	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
21.5.x ✳	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
21.3.x ✳	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
20.9 ✳	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
20.7 ✳	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Windows Enrollment Gateway

Version	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
23.3	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
23.1.0	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Product	Keyfactor Command Compatibility

SQL Server

Version	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
2016	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2017	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2019	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2022	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Windows Server

Version	9.10	9.9	9.8	9.7	9.6	9.5	9.4	9.3	9.2	9.1-9.0
2016	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2019	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2022	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

7.0 Glossary

A

AIA

The authority information access (AIA) is included in a certificate--if configured--and identifies a location from which the chain certificates for that certificate may be retrieved.

AnyAgent

The AnyAgent, one of Keyfactor's suite of orchestrators, is used to allow management of certificates regardless of source or location by allowing customers to implement custom agent functionality via an API.

AnyGateway

The Keyfactor AnyGateway is a generic third party CA gateway framework that allows existing CA gateways and custom CA connections to share the same overall product framework.

API

A set of functions to allow creation of applications. Keyfactor offers the Keyfactor API, which allows third-party software to integrate with the advanced certificate enrollment and management features of Keyfactor Command.

Argument

A parameter or argument is a value that is passed into a function in an application.

Authority Information Access

The authority information access (AIA) is included in a certificate--if configured--and identifies a location from which the chain certificates for that certificate may be retrieved.

B

Bash Orchestrator

The Bash Orchestrator, one of Keyfactor's suite of orchestrators, is used to discover and manage SSH keys across an enterprise.

Blueprint

A snapshot of the certificate stores and scheduled jobs on one orchestrator, which can be used to create matching certificate stores and jobs on another orchestrator with just a few clicks.

C

CA

A certificate authority (CA) is an entity that issues digital certificates. Within Keyfactor Command, a CA may be a Microsoft CA or a Keyfactor gateway to a cloud-based or remote CA.

Certificate Authority

A certificate authority (CA) is an entity that issues digital certificates. Within Keyfactor Command, a CA may be a Microsoft CA or a Keyfactor gateway to a cloud-based or remote CA.

Certificate Revocation List

A Certificate Revocation List (CRL) is a list of digital certificates that have been revoked by the issuing Certificate Authority (CA) before their scheduled expiration date and should no longer be trusted.

Certificate Signing Request

A CSR or certificate signing request is a block of encoded text that is submitted to a CA when enrolling for a certificate. When you generate a CSR within Keyfactor

Command, the matching private key for it is stored in Keyfactor Command in encrypted format and will be married with the certificate once returned from the CA.

CN

A common name (CN) is the component of a distinguished name (DN) that represents the primary name of the object. The value varies depending on the type of object. For a user object, this would be the user's name (e.g. CN=John Smith). For SSL certificates, the CN is typically the fully qualified domain name (FQDN) of the host where the SSL certificate will reside (e.g. servername.keyexample.com or www.keyexample.com).

Collection

The certificate search function allows you to query the Keyfactor Command database for certificates from any available source based on any criteria of the certificates and save the results as a collection that will be availble in other places in the Management Portal (e.g. expiration alerts and certain reports).

Common Name

A common name (CN) is the component of a distinguished name (DN) that represents the primary name of the object. The value varies depending on the type of object. For a user object, this would be the user's name (e.g. CN=John Smith). For SSL certificates, the CN is typically the fully qualified domain name (FQDN) of the host where the SSL certificate will reside (e.g. servername.keyexample.com or www.keyexample.com).

Configuration Tenant

A grouping of CAs. The Microsoft concept of forests is not used in EJBCA so to accommodate the new EJBCA functionality, and to avoid confusion, the term forest needed to be renamed. The new name is configuration tenant. For EJBCA, there would be one configuration tenant per EJBCA server install. For Microsoft, there would be one per forest. Note that configuration tenants cannot be mixed, so Microsoft and EJBCA cannot exist on the same configuration tenant.

CRL

A Certificate Revocation List (CRL) is a list of digital certificates that have been revoked by the issuing Certificate Authority (CA) before their scheduled expiration date and should no longer be trusted.

CSR

A CSR or certificate signing request is a block of encoded text that is submitted to a CA when enrolling for a certificate. When you generate a CSR within Keyfactor Command, the matching private key for it is stored in Keyfactor Command in encrypted format and will be married with the certificate once returned from the CA.

D

DER

A DER format certificate file is a DER-encoded binary certificate. It contains a single certificate and does not support storage of private keys. It sometimes has an extension of .der but is often seen with .cer or .crt.

Distinquished Name

A distinguished name (DN) is the name that uniquely identifies an object in a directory. In the context of Keyfactor Command, this directory is generally Active Directory. A DN is made up of attribute=value pairs,

separated by commas. Any of the attributes defined in the directory schema can be used to make up a DN.

DN

A distinguished name (DN) is the name that uniquely identifies an object in a directory. In the context of Keyfactor Command, this directory is generally Active Directory. A DN is made up of attribute=value pairs, separated by commas. Any of the attributes defined in the directory schema can be used to make up a DN.

DNS

The Domain Name System is a service that translates names into IP addresses.

E

ECC

Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers.

Endpoint

An endpoint is a URL that enables the API to gain access to resources on a server.

Enrollment

Certificate enrollment refers to the process by which a user requests a digital certificate. The user must submit the request to a certificate authority (CA).

EOBO

A user with an enrollment agent certificate can enroll for a certificate on behalf of another user. This is often used when provisioning technology such as smart cards.

F

Forest

An Active Directory forest (AD forest) is the top most logical container in an Active Directory configuration that contains domains, and objects such as users and computers.

G

Gateway Connector

The Keyfactor Gateway Connector is installed in the customer forest to provide a connection between the on-premise CA and the Azure-hosted, Keyfactor managed Hosted Configuration Portal to provide support for synchronization, enrollment and management of certificates through the Azure-hosted instance of Keyfactor Command for the on-premise CA. It is supported on both Windows and Linux.

H

Host Name

The unique identifier that serves as name of a computer. It is sometimes presented as a fully qualified domain name (e.g. servername.keyexample.com) and sometimes just as a short name (e.g. servername).

Hosted Config Portal

The Keyfactor Hosted Configuration Portal is used to configure connections between on-premise instances of the Keyfactor

Gateway Connector and and on-premise CAs to make them available to Azure-hosted instance of Keyfactor Command.The portal is Azure-hosted and managed by Keyfactor.

Hosted Configuration Portal

The Keyfactor Hosted Configuration Portal is used to configure connections between on-premise instances of the Keyfactor Gateway Connector and and on-premise CAs to make them available to Azure-hosted instance of Keyfactor Command.The portal is Azure-hosted and managed by Keyfactor.

Hostname

The unique identifier that serves as name of a computer. It is sometimes presented as a fully qualified domain name (e.g. servername.keyexample.com) and sometimes just as a short name (e.g. servername).

J

Java Agent

The Java Agent, one of Keyfactor's suite of orchestrators, is used to perform discovery of Java keystores and PEM certificate stores, to inventory discovered stores, and to push certificates out to stores as needed.

Java Keystore

A Java KeyStore (JKS) is a file containing security certificates with matching private keys. They are often used by Java-based applications for authentication and encryption.

JKS

A Java KeyStore (JKS) is a file containing security certificates with matching private keys. They are often used by Java-based applications for authentication and encryption.

K

Key Length

The key size or key length is the number of bits in a key used by a cryptographic algorithm.

Key Pair

In asymmetric cryptography, public keys are used together in a key pair with a private key. The private key is retained by the key's creator while the public key is widely distributed to any user or target needing to interact with the holder of the private key.

Key Size

The key size or key length is the number of bits in a key used by a cryptographic algorithm.

Key Type

The key type identifies the type of key to create when creating a symmetric or asymmetric key. It references the signing algorithm and often key size (e.g. AES-256, RSA-2048, Ed25519).

Keyfactor CA Management Gateway

The Keyfactor CA Management Gateway is made up of the Keyfactor Gateway Connector, installed in the customer forest to provide a connection to the local CA, and the Azure-hosted and Keyfactor managed Hosted Configuration Portal. The solution is used to provide a connection between a customer's on-premise CA and an Azure-hosted instance of Keyfactor Command for synchronization, enrollment, and management of certificates.

Keyfactor Gateway Connector

The Keyfactor Gateway Connector is installed in the customer forest to provide a connection between the on-premise CA and the Azure-hosted, Keyfactor managed Hosted Configuration Portal to provide support for synchronization, enrollment and management of certificates through the Azure-hosted instance of Keyfactor Command for the on-premise CA. It is supported on both Windows and Linux.

Keyfactor Universal Orchestrator

The Keyfactor Universal Orchestrator, one of Keyfactor's suite of orchestrators, is used to interact with servers and devices for certificate management, run SSL discovery and management tasks, and manage synchronization of certificate authorities in remote forests. With the addition of custom extensions, it can provide certificate management capabilities on a variety of platforms and devices (e.g. Amazon Web Services (AWS) resources, Citrix\NetScaler devices, F5 devices, IIS stores, JKS keystores, PEM stores, and PKCS#12 stores) and execute tasks outside the standard list of certificate management functions. It runs on either Windows or Linux servers or Linux containers.

Keystore

A Java KeyStore (JKS) is a file containing security certificates with matching private keys. They are often used by Java-based applications for authentication and encryption.

L

Logical Name

The logical name of a CA is the common name given to the CA at the time it is created. For Microsoft CAs, this name can be seen at the top of the Certificate Authority MMC snap-in. It is part of the FQDN\Logical Name string that is used to refer to CAs when using command-line tools and in some Keyfactor Command configuration settings (e.g. ca2.keyexample.com\Corp Issuing CA Two).

M

MAC Agent

The MAC Agent, one of Keyfactor's suite of orchestrators, is used to manage certificates on any keychains on the Mac on which the Keyfactor MAC Agent is installed.

Metadata

Metadata provides information about a piece of data. It is used to summarize basic information about data, which can make working with the data easier. In the context of Keyfactor Command, the certificate metadata feature allows you to create custom metadata fields that allow you to tag certificates with tracking information about certificates.

O

Object Identifier

Object identifiers or OIDs are a standardized system for identifying any object, concept, or "thing" with a globally unambiguous persistent name.

OID

Object identifiers or OIDs are a standardized system for identifying any object, concept, or "thing" with a globally unambiguous persistent name.

Orchestrator

Keyfactor orchestrators perform a variety of functions, including managing certificate stores and SSH key stores.

P

P12

A PFX file (personal information exchange format), also known as a PKCS#12 archive, is a single, password-protected certificate archive that contains both the public and matching private key and, optionally, the certificate chain. It is a common format for Windows servers.

P7B

A PKCS #7 format certificate file is a base64-encoded certificate. Since it's presented in ASCII, you can open it in any text editor. PKCS #7 certificates always begin and end with entries that look something like ---- BEGIN CERTIFICATE---- and ----END CERTIFICATE----. Unlike PEM files, PKCS #7 files can contain only a certificate and its certifiate chain but NOT its private key. Extensions of .p7b or .p7c are usually seen on certificate files of this format.

P7C

A PKCS #7 format certificate file is a base64-encoded certificate. Since it's presented in ASCII, you can open it in any text editor. PKCS #7 certificates always begin and end with entries that look something like ---- BEGIN CERTIFICATE---- and ----END CERTIFICATE----. Unlike PEM files, PKCS #7 files can contain only a certificate and its certifiate chain but NOT its private key. Extensions of .p7b or .p7c are usually seen on certificate files of this format.

Parameter

A parameter or argument is a value that is passed into a function in an application.

PEM

A PEM format certificate file is a base64-encoded certificate. Since it's presented in ASCII, you can open it in any text editor. PEM certificates always begin and end with entries like ---- BEGIN CERTIFICATE---- and ----END CERTIFICATE----. PEM certificates can contain a single certificate or a full certifiate chain and may contain a private key. Usually, extensions of .cer and .crt are certificate files with no private key, .key is a separate private key file, and .pem is both a certificate and private key.

PFX

A PFX file (personal information exchange format), also known as a PKCS#12 archive, is a single, password-protected certificate archive that contains both the public and matching private key and, optionally, the certificate chain. It is a common format for Windows servers.

PKCS #7

A PKCS #7 format certificate file is a base64-encoded certificate. Since it's presented in ASCII, you can open it in any text editor. PKCS #7 certificates always begin and end with entries that look something like ---- BEGIN CERTIFICATE---- and ----END CERTIFICATE----. Unlike PEM files, PKCS #7 files can contain only a certificate and its certifiate chain but NOT its private key. Extensions of .p7b or .p7c are usually seen on certificate files of this format.

PKCS#12

A PFX file (personal information exchange format), also known as a PKCS#12 archive,

is a single, password-protected certificate archive that contains both the public and matching private key and, optionally, the certificate chain. It is a common format for Windows servers.

PKI

A public key infrastructure (PKI) is a set of roles, policies, and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption.

Private Key

Private keys are used in cryptography (symmetric and asymmetric) to encrypt or sign content. In asymmetric cryptography, they are used together in a key pair with a public key. The private or secret key is retained by the key's creator, making it highly secure.

Public Key

In asymmetric cryptography, public keys are used together in a key pair with a private key. The private key is retained by the key's creator while the public key is widely distributed to any user or target needing to interact with the holder of the private key.

Public Key Infrastructure

A public key infrastructure (PKI) is a set of roles, policies, and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption.

R

Rogue Key

A rogue key, in the context of Keyfactor Command, is an SSH public key that appears in an authorized_keys file on a server managed by the SSH orchestrator without authorization.

Root of Trust

A root of trust (RoT) is a source within a cryptographic system that can always be trusted. It is typically a hardened hardware module. HSMs (hardware security modules) and TPMs (trusted platform modules) are examples of RoTs.

RoT

A root of trust (RoT) is a source within a cryptographic system that can always be trusted. It is typically a hardened hardware module. HSMs (hardware security modules) and TPMs (trusted platform modules) are examples of RoTs.

RPC

Remote procedure call (RPC) allows one program to call a function from a program located on another computer on a network without specifying network details. In the context of Keyfactor Command, RPC errors often indicate Kerberos authentication or delegation issues.

rsyslog

Rsyslog is an open-source software utility used on UNIX and Unix-like computer systems for forwarding log messages in an IP network.

S

SAN

The subject alternative name (SAN) is an extension to the X.509 specification that allows you to specify additional values when enrolling for a digital certificate. A variety of SAN formats are supported, with DNS name being the most common.

server name indication

Server name indication (SNI) is an extension to TLS that provides for including the hostname of the target server in the initial handshake request to allow the server to respond with the correct SSL certificate or allow a proxy to forward the request to the appropriate target.

SMTP

Short for simple mail transfer protocol, SMTP is a protocol for sending email messages between servers.

SNI

Server name indication (SNI) is an extension to TLS that provides for including the hostname of the target server in the initial handshake request to allow the server to respond with the correct SSL certificate or allow a proxy to forward the request to the appropriate target.

SSH

The SSH (secure shell) protocol provides for secure connections between computers. It provides several options for authentication, including public key, and protects the communications with strong encryption.

SSL

TLS (Transport Layer Security) and its predecessor SSL (Secure Sockets Layer) are protocols for establishing authenticated and encrypted links between networked computers.

Subject Alternative Name

The subject alternative name (SAN) is an extension to the X.509 specification that allows you to specify additional values when enrolling for a digital certificate. A variety of SAN formats are supported, with DNS name being the most common.

T

Template

A certificate template defines the policies and rules that a CA uses when a request for a certificate is received.

TLS

TLS (Transport Layer Security) and its predecessor SSL (Secure Sockets Layer) are protocols for establishing authenticated and encrypted links between networked computers.

Trusted CA

A certificate authority in the forest in which Keyfactor Command is installed or in a forest in a two-way trust with the forest in which Keyfactor Command is installed.

U

Untrusted CA

A certificate authority in a forest in a one-way trust with the forest in which Keyfactor Command is installed or in a forest that is untrusted by the forest in which Keyfactor Command is installed. Non-domain-joined standalone CAs also fall into this category.

W

Web API

A set of functions to allow creation of applications. Keyfactor offers the Keyfactor API, which allows third-party software to integrate with the advanced certificate enrollment and management features of Keyfactor Command.

Windows Orchestrator

The Windows Orchestrator, one of Keyfactor's suite of orchestrators, is used to manage synchronization of certificate authorities in remote forests, run SSL discovery and management tasks, and interact with Windows servers as well as F5 devices, NetScaler devices, Amazon Web Services (AWS) resources, and FTP capable devices, for certificate management. In addition, the AnyAgent capability of the Windows Orchestrator allows it to be extended to create custom certificate store types and management capabilities regardless of source platform or location.

Workflow

A workflow is a series of steps necessary to complete a process. In the context of Keyfactor Command, it refers to the workflow builder, which allows you automate event-driven tasks when a certificate is requested or revoked.

X

x.509

In cryptography, X.509 is a standard defining the format of public key certificates. An X.509 certificate contains a public key and an identity (e.g. a host name or an organization or individual name), and is either signed by a certificate authority or self-signed. When a certificate is signed by a trusted certificate authority it can be used to establish trusted secure communications with the owner of the corresponding private key. It can also be used to verify digitally signed documents and emails.

8.0 Copyright Notice