KEYFACTOR

2023 | Keyfactor

Keyfactor Web APIs 11.1

Reference Guide

Table of Contents

List of Tables

List of Figures

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 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.

2.1 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.

2.2 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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). 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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*).

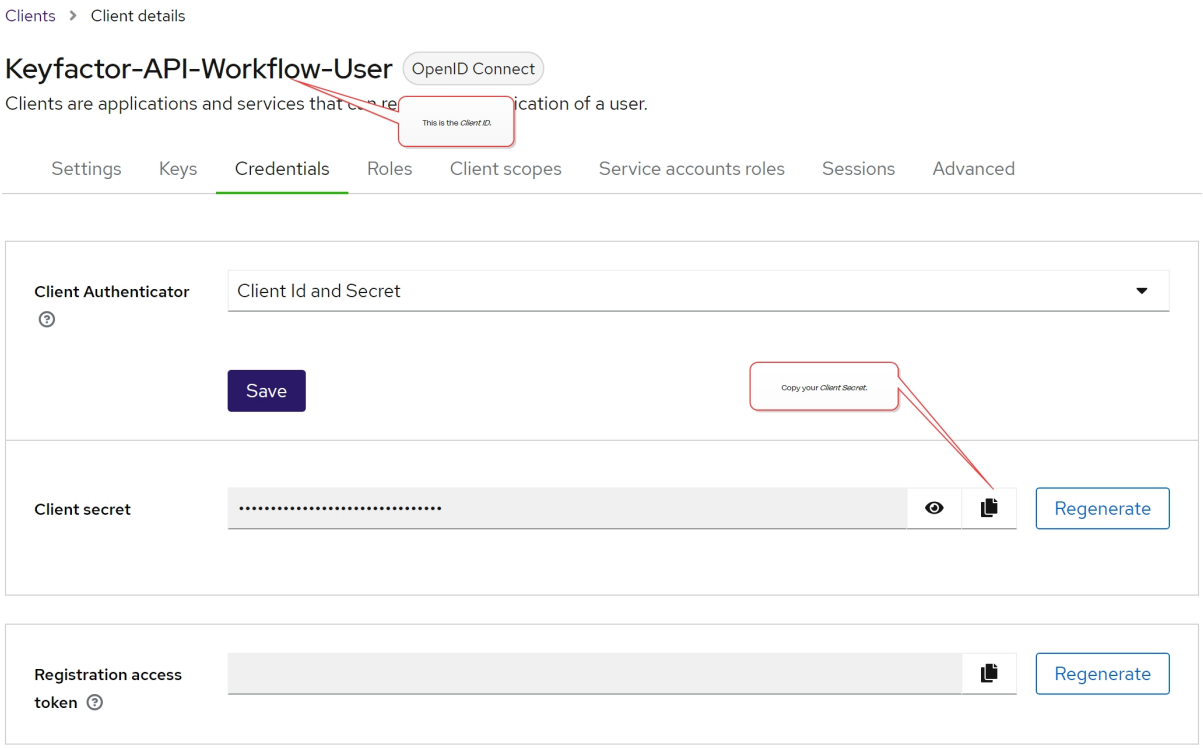


Figure 1: Client Secret for Keyfactor API in Keyfactor Identity Provider

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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*) 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/Keyfactor/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 2: 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.

Figure 2: 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.

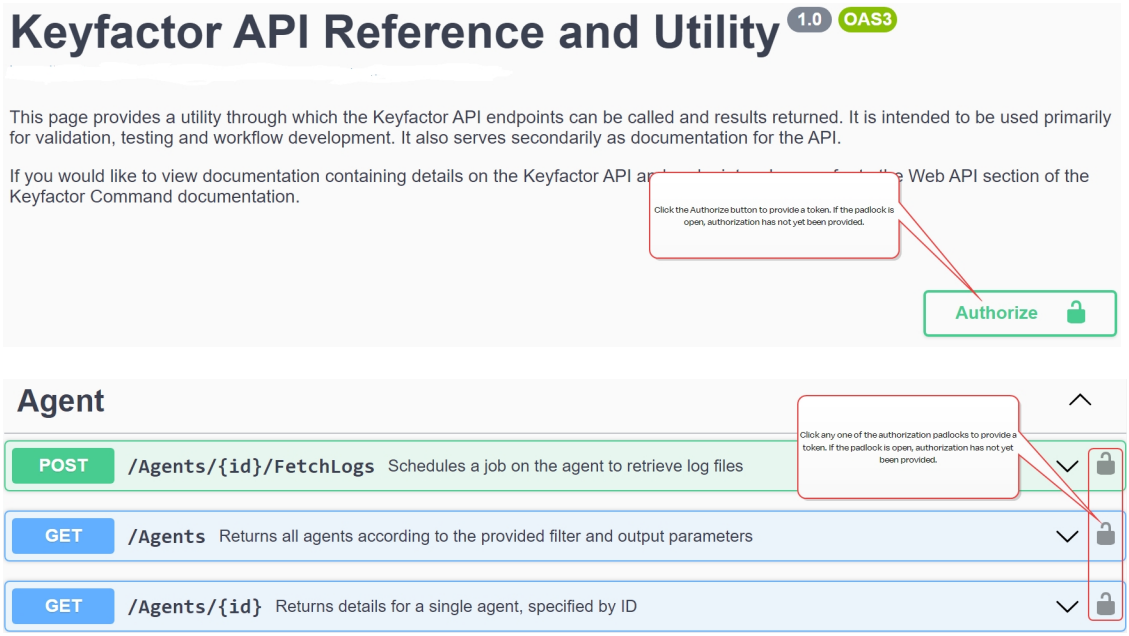


Figure 3: Keyfactor API Reference and Utility Authorize Options

3. In the Available authorizations dialog, paste in your access token value and click **Authorize**.

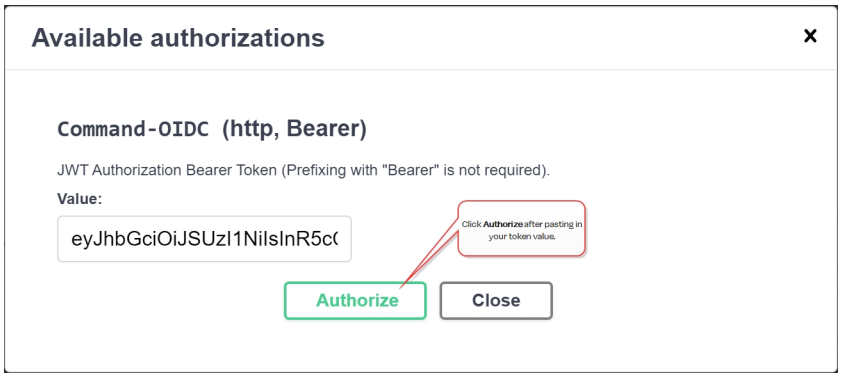


Figure 4: Enter Access Token in the Keyfactor API Reference and Utility

4. If authorization is successful, the Authorize button will change to *Logout*, and the padlocks will change to locked.

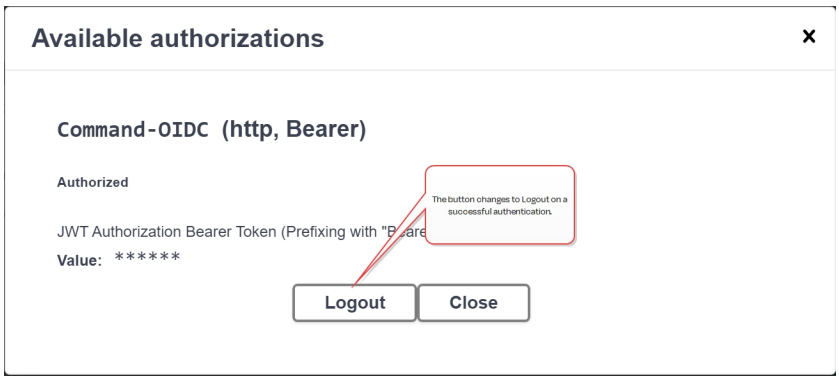


Figure 5: Successful Authorization in the Keyfactor API Reference and Utility

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 *Workflow Definitions Configuration Parameters Example: Use Custom PowerShell with Embedded REST Request, Send Email, and Require Approval* in the *Keyfactor API Reference Guide*.

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.

2.3 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 in the *Keyfactor Command Reference Guide*). 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.

2.4 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 1: Common Request Headers, common response headers (for successful requests and certain unsuccessful requests) are given in Table 2: Common Response Headers, and HTTP statuses are given in Table 3: 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 1: 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 2: 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 3: 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.

2.5 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 6: 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.

2.6 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*.

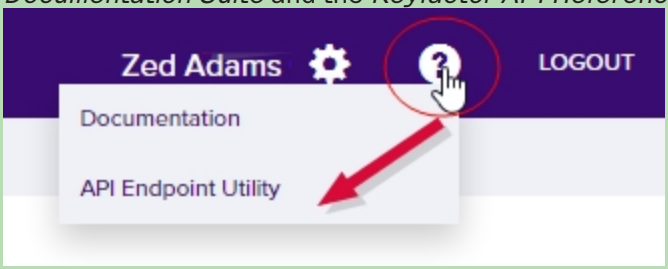


Figure 7: Documentation in the Help Dropdown

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.

2.6.1 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 4: 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 17
/Reset	POST	Resets one or more orchestrators or agents to a new state and clears jobs.	POST Agents Reset on page 23

Endpoint	Method	Description	Link
/Approve	POST	Approves an orchestrator.	POST Agents Approve on page 24
/Disapprove	POST	Disapproves an orchestrator.	POST Agents Disapprove on page 24
/{id}/Reset	POST	Resets a single orchestrator or agent to a new state and clears jobs.	POST Agents ID Reset on page 25
/{id}/FetchLogs	POST	Schedules a job on the orchestrator or agent to retrieve log files.	POST Agents ID FetchLogs on page 26
/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 26

2.6.1.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 5: 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 17) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Table 6: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.1.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 7: 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* in the *Keyfactor Command Reference Guide*. The query fields supported for this endpoint are: • AgentId • Blueprint • Capabilities (See Table 8: 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 8: 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

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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.1.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 9: 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 17) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.1.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 10: 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 17) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.1.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 11: 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 17) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.1.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 12: 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 17) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.1.7 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* in the *Keyfactor Command Reference Guide*) 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 1021).

Table 13: 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 17) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.1.8 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 14: 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 17) 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 15: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.2 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 16: Agent BluePrint Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the orchestrator blueprint	DELETE Agent

Endpoint	Method	Description	Link
		with the specified GUID.	BluePrint ID on the 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 31
/{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 32
/{id}/Stores	GET	Returns details of the certificate stores for the orchestrator blueprint with the specified GUID.	GET Agent BluePrint ID Stores on page 37
/ApplyBlueprint	POST	Applies an orchestrator blueprint to one or more orchestrators.	POST AgentBluePrint ApplyBluePrint on page 40
/GenerateBlueprint	POST	Creates a new orchestrator blueprint from an orchestrator.	POST AgentBluePrint GenerateBluePrint on page 41

2.6.2.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 17: 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 31) to retrieve a list of all the blueprints to determine the orchestrator blueprint GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.2.2 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 32) or GET /AgentBluePrint/{id}/Stores method (see GET Agent BluePrint ID Stores on page 37).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 18: 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 19: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.2.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 20: 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 21: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.2.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 22: 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 31) 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 23: 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* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* 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 643).</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 713 for more information).</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.2.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 24: 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 31) 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 25: 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* 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 643).
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 713 for more information).

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.2.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 26: 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 31) 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 17) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.2.7 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/
/agents/management/modify/

Table 27: 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 17) 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 28: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.3 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 29: 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 45
/	POST	Creates an orchestrator pool based on information in the request.	POST Agent Pools on page 48
/	PUT	Updates an orchestrator pool based on information in the request.	PUT Agent Pools on page 50
/Agents	GET	Returns a list of orchestrators associated with the Default Agent Pool.	GET Agent Pools Agents on page 53

2.6.3.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/
/ssl/modify/

Table 30: 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 45) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.3.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 31: 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 45) to retrieve a list of all the orchestrator pools to determine the orchestrator pool GUID.

Table 32: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.3.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 33: 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* in the *Keyfactor Command Reference Guide*. 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 34: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.3.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/
/ssl/modify/

Table 35: 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 36: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.3.5 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* in the *Keyfactor Command Reference Guide*) 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 37: 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 38: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.3.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 39: 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* in the *Keyfactor Command Reference Guide*. 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 40: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.4 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 88
- Alerts Issued on page 126
- Alerts Key Rotation on page 163
- Alerts Pending on page 196

2.6.4.1 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 41: 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 78
/Alerts/Denied	GET	Retrieves details for all configured denied certificate request alerts.	GET Alerts Denied on page 62
/Alerts/Denied	POST	Creates a new denied certificate request alert.	POST Alerts Denied on page 68

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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 42: 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 62) to retrieve a list of all the issued request alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 43: 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 62) to retrieve a list of all the issued request alerts to determine the alert ID.

Table 44: 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: Substitutable Special Text for Denied Certificate Request 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 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Denied Certificate Request Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 45: 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* in the *Keyfactor Command Reference Guide*. 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 46: 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: Substitutable Special Text for Denied Certificate Request 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 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Denied Certificate Request Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 47: 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: {caregid}</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: Substitutable Special Text for Denied Certificate Request 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 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 1593) 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Denied Certificate Request Alerts* in the *Keyfactor Command Reference Guide* 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:

Name	In	Description

		"EventHandlerParameters": [
		{
		"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 48: 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: Substitutable Special Text for Denied Certificate Request 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 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Denied Certificate Request Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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 49: 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: Substitutable Special Text for Denied Certificate Request 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 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 1593) 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Denied Certificate Request Alerts* in the *Keyfactor Command Reference Guide* 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:

Name	In	Description

		"EventHandlerParameters": [
		{
		"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 50: 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: Substitutable Special Text for Denied Certificate Request 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 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Denied Certificate Request Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.4.2 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 51: 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 94
/Alerts/Expiration/Schedule	PUT	Updates the schedule for delivery of expired certificate alerts.	PUT Alerts Expiration Schedule on page 95
/Alerts/Expiration	GET	Retrieves details for all configured expired certificate alerts.	GET Alerts Expiration on page 96
/Alerts/Expiration	POST	Creates a new expired certificate alert.	POST Alerts Expiration on page 102
/Alerts/Expiration	PUT	Updates an expired certificate for the specified ID.	PUT Alerts Expiration on page 112
/Alerts/Expiration/Test	POST	Test an Expiration Alert	POST Alerts Expiration Test on page 122
/Alerts/Expiration/TestAll	POST	Test All Expiration Alerts	POST Alerts Expiration Test All on page 124

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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 52: 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 96) to retrieve a list of all the expiration alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 53: 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 96) to retrieve a list of all the expiration alerts to determine the alert ID.

Table 54: 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* 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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>2nd at 12:00 am UTC would be alerted on.</td></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 in the Keyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 55: GET 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 56: 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: **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 57: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 58: 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* in the *Keyfactor Command Reference Guide*. 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 59: 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* 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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>2nd at 12:00 am UTC would be alerted on.</td></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 in the Keyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 60: 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: Substitutable Special Text for Expiration 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.
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.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
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 461) 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 Using Event Handlers in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). - 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* for a </td></tr></table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>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": "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 61: 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* 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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>2nd at 12:00 am UTC would be alerted on.</td></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 in the Keyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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 62: 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: Substitutable Special Text for Expiration 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.
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 461) 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 Using Event Handlers in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). - 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* for a </td></tr></table>

Name	In	Description

		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>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": "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 63: 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* 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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>2nd at 12:00 am UTC would be alerted on.</td></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 in the Keyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Expiration Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 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 regardless of the setting of the *SendAlerts* flag.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 64: 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 96) 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 65: POST Alerts Expiration Test 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: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 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 regardless of the setting of the *SendAlerts* flag.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 66: 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 67: 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: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.4.3 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 68: 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 132
/Alerts/Issued/Schedule	PUT	Updates the schedule for delivery of issued certificate request alerts.	PUT Alerts Issued Schedule on page 134
/Alerts/Issued	GET	Retrieves details for all configured issued certificate request alerts.	GET Alerts Issued on page 137
/Alerts/Issued	POST	Creates a new issued certificate request alert.	POST Alerts Issued on page 143
/Alerts/Issued	PUT	Updates an issued certificate request alert for the specified ID.	PUT Alerts Issued on page 153

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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 69: 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 137) to retrieve a list of all the issued request alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 70: 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 137) to retrieve a list of all the issued request alerts to determine the alert ID.

Table 71: 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: 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Issued Certificate Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 72: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 73: 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:

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"
}
```

Table 74: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 75: 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* in the *Keyfactor Command Reference Guide*. 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 76: 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: 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Issued Certificate Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 77: 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: 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 1593) 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Issued Certificate Alerts* in the *Keyfactor Command Reference Guide* 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:

Name	In	Description

		"EventHandlerParameters": [
		{
		"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 78: 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: 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Issued Certificate Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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 79: 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: 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 1593) 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Issued Certificate Alerts* in the *Keyfactor Command Reference Guide* 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:

Name	In	Description

		"EventHandlerParameters": [
		{
		"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 80: 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: 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Issued Certificate Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.4.4 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* in the *Keyfactor Command Reference Guide*). Key rotation alerts apply to both user keys (see *My SSH Key* in the *Keyfactor Command Reference Guide*) and service account keys (see *Service Account Keys* in the *Keyfactor Command Reference Guide*) generated within Keyfactor Command.

Table 81: 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 on the next page
/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 167
/Alerts/KeyRotation/Schedule	PUT	Updates the schedule for delivery of SSH key rotation alerts.	PUT Alerts Key Rotation Schedule on page 169
/Alerts/KeyRotation	GET	Retrieves details for all configured SSH key rotation alerts.	GET Alerts Key Rotation on page 172
/Alerts/KeyRotation	POST	Creates a new SSH key rotation alert.	POST Alerts Key Rotation on page 176
/Alerts/KeyRotation	PUT	Updates the SSH key rotation alert for a specified ID.	PUT Alerts Key Rotation on page 184
/Alerts/KeyRotation/Test	POST	Used to test specific SSH key rotation alerts.	POST Alerts Key Rotation Test on page 192
/Alerts/KeyRotation/TestAll	POST	Used to test all SSH key rotation alerts.	POST Alerts Key Rotation Test All on page 194

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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 82: 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 172) to retrieve a list of all the SSH key rotation alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 83: 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 172) to retrieve a list of all the SSH key rotation alerts to determine the alert ID.

Table 84: 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 85: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 86: 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:

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"
}
```

Table 87: 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:
	
	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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 88: 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* in the *Keyfactor Command Reference Guide*. 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 89: 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:

/monitoring/alerts/modify/

Table 90: 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* for a complete list of available substitutable special text strings.
Rota-	Body	An integer indicating the number of days prior to the end of an SSH key's

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>tionWarningDays</td><td></td><td>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>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 in the Keyfactor Command Reference Guide.</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</td></tr></table></td></tr>
</table>

Name	In	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* in the *Keyfactor API Reference Guide*).
		• 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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"
		  },
		  {</pre>

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 91: 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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 92: 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* for a complete list of available

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>substitutable special text strings.</td></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 in the Keyfactor Command Reference Guide.</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</td></tr></table></td></tr>
</table>

Name	In	Description

Value	Description
	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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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": [
  {
```

Name	In	Description

		"Id": 28,
		"Key": "user",
		"DefaultValue": "username",
		"ParameterType": "Token"
		},
		{
		"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 93: 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Key Rotation Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 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 regardless of the setting of the *SendAlerts* flag.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 94: 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 172) 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 95: 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: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 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 regardless of the setting of the *SendAlerts* flag.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 96: 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 97: POST Alerts Key Rotation Test All Response Data

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>KeyRotationAlerts</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>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: Substitutable Special Text for Key Rotation Alerts in the Keyfactor Command Reference Guide for a complete list of available substitutable special text strings.</td></tr>
<tr><td>Recipient</td><td>A string indicating the recipient for the alert.</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.4.5 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 1659).
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* in the *Keyfactor Command Reference Guide*.

Table 98: 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 222
/Alerts/Pending/Schedule	GET	Retrieves details of the schedule for delivery of pending certificate request alerts.	GET Alerts Pending Schedule on page 202
/Alerts/Pending/Schedule	PUT	Updates the schedule for delivery of pending certificate request alerts.	PUT Alerts Pending Schedule on page 204
/Alerts/Pending	GET	Retrieves details for all configured pending certificate request alerts.	GET Alerts Pending on page 207
/Alerts/Pending	POST	Creates a new pending certificate request alert.	POST Alerts Pending on page 212
/Alerts/Pending/Test	POST	Tests all alerts	POST Alerts Pending TestAll on page 235
/Alerts/Pending/Test/{id}	POST	Tests specific alerts	POST Alerts Pending Test on page 233

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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 99: 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 207) to retrieve a list of all the pending request alerts to determine the alert ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 100: 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 207) to retrieve a list of all the pending request alerts to determine the alert ID.

Table 101: 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: Substitutable Special Text for Pending Request 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 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 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Pending Request Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 102: GET 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 103: PUT Alerts Pending Schedule Input Parameters

Name	In	Description

Schedule	Body	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" }`

Table 104: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 105: 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* in the *Keyfactor Command Reference Guide*. 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 106: 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: Substitutable Special Text for Pending Request 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 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 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Pending Request Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 107: 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: Substitutable Special Text for Pending Request 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 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 1593) 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 in the Keyfactor Command Reference Guide.</td></tr>
<tr><td>EventHand-
lerParameters</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 in the Keyfactor API Reference Guide).
• 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

Value	Description
	script. See *Table: Substitutable Special Text for Pending Request Alerts* in the *Keyfactor Command Reference Guide* 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": "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"
   }
]
```

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 108: 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: Substitutable Special Text for Pending Request 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 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 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*). • 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: Substitutable Special Text for Pending Request Alerts* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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 109: 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: Substitutable Special Text for Pending Request 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 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 1593) 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 in the Keyfactor Command Reference Guide.</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* in the *Keyfactor API Reference Guide*).
		• 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: Substitutable Special Text for Pending Request Alerts* in the *Keyfactor Command Reference Guide* 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"
		  },
		  {</pre>

Name	In	Description

		"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"
		}
		]

Table 110: 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: Substitutable Special Text for Pending Request 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 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 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 in the Keyfactor Command Reference Guide.</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 in the Keyfactor API Reference Guide).
• 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: Substitutable Special Text for Pending Request Alerts in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 in the *Keyfactor Command Reference Guide*). 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 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) 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 111: 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 112: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 in the *Keyfactor Command Reference Guide*). 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 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) 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 113: 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 114: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.5 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 in the *Keyfactor Command Reference Guide*.

Table 115: 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 241
/{id}	GET	Returns details for a single application setting.	GET AppSetting ID on page 239
/{id}/Set	PUT	Updates the value configured for an application setting based on its reference ID.	PUT AppSetting ID Set on page 243
/{name}/Set	PUT	Updates the value configured for an application setting based on its reference name.	PUT AppSetting Name Set on page 245

2.6.5.1 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:

/application_settings/read/

Table 116: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.5.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/application_settings/read/

Table 117: 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 17) to retrieve a list of all the application settings to determine the application setting ID.

Table 118: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.5.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/application_settings/read/
/application_settings/modify/

Table 119: 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 237) 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 120: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.5.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/application_settings/read/
/application_settings/modify/

Table 121: 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 237) 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 122: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.5.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/application_settings/read/
/application_settings/modify/

Table 123: 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 237) 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 124: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.6 Audit

The Audit component of the Keyfactor API is used to track changes to the Keyfactor Command operation and configuration.

Table 125: 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 253
/	GET	Returns a list of all audit log entries according to the provided filters and input parameters.	GET Audit on page 254
/Download	GET	Returns a comma separated list of audit log entries according to the provided filters and input parameters.	GET Audit Download on page 260
/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 264

2.6.6.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/auditing/read/

Table 126: 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 254) to retrieve a list of all the audit log entries to determine the audit log entry ID.

Table 127: 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

<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>

<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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.6.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/auditing/read/

Table 128: 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 129: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.6.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/auditing/read/

Table 130: 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* in the *Keyfactor Command Reference Guide*. The query fields supported for this endpoint are: • Name (EntityIdentifier) • Category (EntityType) (see Table 131: GET Audit Response Data for codes) • ImmutableIdentifier • Level (see Table 131: GET Audit Response Data for codes) • Operation (see Table 131: 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 131: 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>

<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>

<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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.6.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/auditing/read/

Table 132: 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* in the *Keyfactor Command Reference Guide*. The query fields supported for this endpoint are: • Name (EntityIdentifier) • Category (EntityType) (see Table 131: GET Audit Response Data for codes) • ImmutableIdentifier • Level (see Table 131: GET Audit Response Data for codes) • Operation (see Table 131: 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 133: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.6.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/auditing/read/

Table 134: 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* in the *Keyfactor Command Reference Guide*. The query fields supported for this endpoint are: • Name (EntityIdentifier) • Category (EntityType) (see Table 131: GET Audit Response Data for codes) • ImmutableIdentifier • Level (see Table 131: GET Audit Response Data for codes) • Operation (see Table 131: 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 135: 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

<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:

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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7 Certificates

The Certificates component of the Keyfactor API supports certificate lifecycle and management tasks, apart from enrollment.

Table 136: 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 272
/{id}/Validate	GET	Validates that a certificate chain	GET Certificates ID Validate on page 274

Endpoint	Method	Description	Link
		can be built for the specified certificate.	
/Locations/{id}	GET	Returns details about the certificates stores in which the certificate is located.	GET Certificates Locations ID on page 279
/IdentityAudit/{id}	GET	Returns audit identity permissions for certificate.	GET Certificates Identity Audit ID on page 282
/{id}	DELETE	Deletes a certificate from the Keyfactor Command database by its ID.	DELETE Certificates ID on page 286
/{id}	GET	Returns certificate details for a specified certificate.	GET Certificates ID on page 287
/Metadata/Compare	GET	Compares the metadata value provided with the metadata value associated with the specified certificate.	GET Certificates Metadata Compare on page 300
/{id}/History	GET	Returns the certificate operations history for a specified certificate.	GET Certificates ID History on page 301
/	DELETE	Deletes multiple certificates from the Keyfactor Command database, as specified by the IDs in the request body.	DELETE Certificates on page 303
/	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 305
/Metadata	PUT	Updates the metadata for a specified certificate.	PUT Certificates Metadata on page 320
/Metadata/All	PUT	Updates the metadata for an array of certificate IDs.	PUT Certificates Metadata All on page 321
/Import	POST	Imports a certificate into	POST Certificates

Endpoint	Method	Description	Link
		Keyfactor Command.	Import on page 326
/Revoke	POST	Revokes a certificate.	POST Certificates Revoke on page 330
/Analyze	POST	Reads a base-64 encoded PEM certificates and returns it in human-readable form.	POST Certificates Analyze on page 332
/Recover	POST	Returns a recovered certificate as a PFX.	POST Certificates Recover on page 333
/Download	POST	Downloads a certificate.	POST Certificates Download on page 337
/RevokeAll	POST	Revokes all the certificates in the provided query.	POST Certificates Revoke All on page 340
/Query	DELETE	Deletes multiple certificates from the Keyfactor Command database based on search query.	DELETE Certificates Query on page 343
/PrivateKey	DELETE	Deletes the stored private keys of multiple certificates within the Keyfactor Command database.	DELETE Certificates Private Key on page 344
/PrivateKey/{id}	DELETE	Deletes the stored private key(s) of a certificate within the Keyfactor Command database.	DELETE Certificates Private Key ID on page 345

2.6.7.1 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 137: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Table 138: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.2 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 139: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Table 140: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.3 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 141: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Table 142: 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 713) 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* for more information.

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>StorePath</td><td>A string containing 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 Certificate Store Operations: Adding or Modifying a Certificate Store in the Keyfactor Command Reference Guide for more information.</td></tr><tr><td>Alias</td><td>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 in the Keyfactor Command Reference Guide for more information.</td></tr></table></td></tr></table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.4 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* 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 11.

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 143: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Table 144: 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
	Parameter / **Description** GrantedBy — A array of strings containing the list of roles or collections that grant the given permission to the entity.

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 145: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Table 146: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.5 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* 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 147: 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 305) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.6 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 148: 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 305) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Table 149: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.7 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 150: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.8 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* in the *Keyfactor Command Reference Guide*.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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 151: 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 Permissions* in the *Keyfactor Command Reference Guide* 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 152: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.9 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* 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 153: 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 on the next page) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.10 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 154: 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 Permissions* in the *Keyfactor Command Reference Guide* 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*. The query fields supported for this endpoint are: • ArchivedKey • CertId • CA

Name	In	Description
		• CertState • 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*

Name	In	Description
		• *CertRequestId* • *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 155: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.11 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 156: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.12 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 157: 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* in the *Keyfactor Command Reference Guide*. 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) • EKU • EKUName • HasPrivateKey • ImportDate • IssuedDate (aliases: NotBefore and EffectiveDate) • IssuerDN • KeySize (alias: KeySizeInBits) • KeyType • KeyUsage

Name	In	Description

		• 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).
		Tip: To exclude revoked certificates from the update, include a query of:
		`CertState -ne \"2\"`
		To exclude expired certificates from the update, include a query of:
		`ExpirationDate -ge \"%TODAY%\"`

CertificateIds	Body	**Required***. An array of integers indicating the Keyfactor Command certi-

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>ficate 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": [
   {
      "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,</pre></td></tr>
</table>

Name	In	Description

		<pre>"OverwriteExisting": true
		},
		{
		"MetadataName": "Notes", // This is a BigText field.
		"Value": "Here are some notes about this certificate.",
		"OverwriteExisting": true
		},
		{
		"MetadataName": "BbusinessUnit", // 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
		}
		]</pre>

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 Permissions* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.13 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/import/

Table 158: 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* in the *Keyfactor Command Reference Guide* 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

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 159: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.14 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide*).

Table 160: 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 Certificate Permissions in the Keyfactor Command Reference Guide for more information.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.15 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 161: 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 162: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.16 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 337).

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 *Certificate Template Operations: Details Tab* in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 163: 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 Permissions* in the *Keyfactor Command Reference Guide* 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 164: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.17 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 333).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 165: 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 Permissions* in the *Keyfactor Command Reference Guide* 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 166: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.18 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 in the *Keyfactor Command Reference Guide*). 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide*).

Table 167: 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 in the Keyfactor Command Reference Guide. 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>
</table>

Name	In	Description
IncludeExpired	Body	A Boolean that indicates whether expired certificates should be included in 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 Permissions* in the *Keyfactor Command Reference Guide* 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.

2.6.7.19 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 168: 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* in the *Keyfactor Command Reference Guide* 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.20 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 169: 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 305) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.7.21 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the *CollectionId* input parameter, below.

Table 170: 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 305) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8 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 171: 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 449
/	GET	Returns a list of all	GET Certificate

Endpoint	Method	Description	Link
		certificate authorities.	Authority on page 365
/	POST	Creates a new certificate authority record.	POST Certificate Authority on page 381
/	PUT	Updates an existing certificate authority record.	PUT Certificate Authority on page 413
/{id}	GET	Returns details for the certificate authority identified by the specified ID.	GET Certificate Authority ID on page 350
/{id}	DELETE	Deletes the certificate authority record for the specified ID.	DELETE Certificate Authority ID on page 349
/Test	POST	Validates that the certificate authority with the provided information can be reached.	POST Certificate Authority Test on page 445
/SourceCount	GET	Retrieve the count of certificate authorities with full or incremental synchronization scans configured.	GET Certificate Authority Source Count on page 449
/AvailableForests	GET	Retrieve the list of forests in Active Directory Keyfactor Command	GET Certificate Authority Available Forests on page 450
/Import	POST	Import into Keyfactor Command any certificate authorities from the provided configuration	POST Certificate Authority Import on page 459

Endpoint	Method	Description	Link
		tenant DNS suffix (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 451
/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 452
/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 452
/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 453
/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 455
/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 454
/AlertRecipients/CAThresholdRecipients	POST	Create new recip-	POST Certificate

Endpoint	Method	Description	Link
		ients to receive CA threshold alerts in Keyfactor Command.	Authority Alert Recipients CA Threshold Recipients on page 458
/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 456
/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 457
/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 459
/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 456

2.6.8.1 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* in the *Keyfactor Command Reference Guide*) 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 172: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 173: 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 174: 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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

Name	Description
	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	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* in the *Keyfactor Command Reference Guide* 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

Name	Description
	notification will be included in the threshold monitoring email. This value is unset by default.
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 1606).
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 1606). 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 Certificate Authority Operations: Adding or Modifying a CA Record in the Keyfactor Command Reference Guide 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 Expli-</td></tr>
</table>

Name	Description
	citPassword 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	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* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide*) 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

Name	Description
	Keyfactor Command.
ExplicitPassword	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)* in the *Keyfactor Command Reference Guide* 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 *Configuring Template Options* in the *Keyfactor Command Reference Guide* and see PUT Templates on page 1606).
AllowedRequesters	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For

Name	Description
	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 1606). 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:

<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": {
   "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 Certificate Template Operations and Certificate Authority Operations in the Keyfactor Command Reference Guide).</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 175: 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 176: 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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

Name	Description
	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	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* in the *Keyfactor Command Reference Guide* 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

Name	Description
	notification will be included in the threshold monitoring email. This value is unset by default.
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 1606).
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 1606). 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 Certificate Authority Operations: Adding or Modifying a CA Record in the Keyfactor Command Reference Guide 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 Expli-</td></tr>
</table>

Name	Description
	citPassword 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	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* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide*) 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

Name	Description
	Keyfactor Command.
ExplicitPassword	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)* in the *Keyfactor Command Reference Guide* 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 *Configuring Template Options* in the *Keyfactor Command Reference Guide* and see PUT Templates on page 1606).
AllowedRequesters	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For

Name	Description
	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 1606). 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:

<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": {
   "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 Certificate Template Operations and Certificate Authority Operations in the Keyfactor Command Reference Guide).</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/modify/

Table 177: 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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).

Name	In	Description
		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* in the *Keyfactor Command Reference Guide* for more information.

Name	In	Description
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 1606).
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

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td>y</td><td>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 1606).
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 Certificate Authority Operations: Adding or Modifying a CA Record in the Keyfactor Command Reference Guide 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>
</table>

Name	In	Description
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* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide*) and the *AllowedRequesters* option.

Name	In	Description
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)* in the *Keyfactor Command Reference Guide* 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 *Configuring Template Options* in the *Keyfactor Command Reference Guide* and see PUT Templates on page 1606).
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:

Name	In	Description

		<pre>"AllowedRequesters": [
		  "Power Users",
		  "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 1606).
		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 were created via the API. The following schedule types are supported:

<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>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>
<tr><td>Week-</td><td>A dictionary that indicates a job scheduled to run on 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>ly</td><td>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>
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>"FullScan": {
		  "Daily": {
		  "Time": "2022-05-27T17:30:00Z"
		  }
		}</pre>
		Or:
		<pre>"FullScan": {
		  "Weekly": {
		  "Days": [
		  "Monday",
		  "Wednesday",
		  "Friday"
		  ],
		  "Time": "2022-05-27T17:30:00Z"
		  }
		}</pre>
		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.

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

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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:
<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</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>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>
<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": {</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>

```
    "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:

```
"Interval": {
```

</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>   "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)* in the *Keyfactor Command Reference Guide* and PAM Providers on page 1038 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 1065) 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 1065 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 1065 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* and *Certificate Authority Operations* in the *Keyfactor Command Reference Guide*).
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 178: 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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

Name	Description
	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	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* in the *Keyfactor Command Reference Guide* 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

Name	Description
	notification will be included in the threshold monitoring email. This value is unset by default.
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 1606).
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 1606). 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 Certificate Authority Operations: Adding or Modifying a CA Record in the Keyfactor Command Reference Guide 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 Expli-</td></tr>
</table>

Name	Description
	citPassword 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	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* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide*) 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

Name	Description
	Keyfactor Command.
ExplicitPassword	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)* in the *Keyfactor Command Reference Guide* 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 *Configuring Template Options* in the *Keyfactor Command Reference Guide* and see PUT Templates on page 1606).
AllowedRequesters	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For

Name	Description
	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 1606). 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

	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:
	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><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 Certificate Template Operations and Certificate Authority Operations in the Keyfactor Command Reference Guide).</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.5 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* in the *Keyfactor Command Reference Guide*) 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 179: 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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*

Name	In	Description
		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 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

Name	In	Description
		also configure email recipients for threshold alerts. These are configured globally rather than on a CA-by-CA basis. See *Certificate Authority Monitoring* in the *Keyfactor Command Reference Guide* 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 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 1606).
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 certi-

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>ficates 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:
<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 1606).
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 Certificate Authority Operations: Adding</td></tr>
</table>

Name	In	Description
		or Modifying a CA Record in the *Keyfactor Command Reference Guide* for more information.
KeyRetentionDays	Body	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.
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* in the *Keyfactor Command Reference Guide* 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

Name	In	Description
		• 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* in the *Keyfactor Command Reference Guide*) 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)* in the *Keyfactor Command Reference Guide* 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,

Name	In	Description
		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 *Configuring Template Options* in the *Keyfactor Command Reference Guide* and see PUT Templates on page 1606).
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 1606). The values set here are only considered if *UseAllowedRequesters* is set to *true*.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>FullScan</td><td>Body</td><td>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:
<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>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: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>
<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:
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description

Nam-e	Description

```
"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.

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

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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.</td></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></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>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: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 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:</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>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: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—
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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.
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) in the Keyfactor Command Reference Guide and <a>PAM Providers on page 1038 for more information.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SecretValue</td><td>A string containing the password used to security the EJBCA CA client authentication certificate.</td></tr>
<tr><td>Parameters</td><td>An object 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 1065) to retrieve a list of all the PAM providers to determine the ID.</td></tr>
</table>
For example, the password stored as a Keyfactor secret will look like:</td></tr>
</table>

Name	In	Description

		<pre>{
		"SecretValue": "MySuperSecretPassword"
		}</pre>
		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 1065 and the Folder and Object reference the folder name and object name in the CyberArk safe):
		<pre>{
		"Provider": "1",
		"Parameters":{
		"Folder":"MyFolderName",
		"Object":"MyEJBCAClientAuthPassword"
		}
		}</pre>
		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 1065 and the SecretId is the ID if the secret created in the Delinea secret server for this purpose):
		<pre>{
		"Provider": "1",
		"Parameters":{
		"SecretId":"MyEJBCAPasswordId"
		}
		}</pre>
		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:
		<pre>{
		"SecretValue": "MIACAQMwgAYJKoZIhvcNAQcBoIAkgASCA+[trun-</pre>

Name	In	Description
		cated 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.
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* and *Certificate Authority Operations* in the *Keyfactor Command Reference Guide*).
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 180: 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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 *Certificate Authority Operations: Adding or Modifying a CA Record* in the *Keyfactor Command Reference Guide* 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

Name	Description
	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	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* in the *Keyfactor Command Reference Guide* 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

Name	Description
	notification will be included in the threshold monitoring email. This value is unset by default.
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 1606).
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 1606). 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 Certificate Authority Operations: Adding or Modifying a CA Record in the Keyfactor Command Reference Guide 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 Expli-</td></tr>
</table>

Name	Description
	citPassword 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	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* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide*) 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

Name	Description
	Keyfactor Command.
ExplicitPassword	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)* in the *Keyfactor Command Reference Guide* 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 *Configuring Template Options* in the *Keyfactor Command Reference Guide* and see PUT Templates on page 1606).
AllowedRequesters	An array of strings indicating Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For

Name	Description
	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 1606). 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

	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:
	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: Time Description: 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).
	Name: Days Description: 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 Certificate Template Operations and Certificate Authority Operations in the Keyfactor Command Reference Guide).</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 181: 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* in the *Keyfactor Command Reference Guide*) 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 182: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.7 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 183: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.8 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 184: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.9 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 185: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.10 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 186: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.11 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 187: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.12 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/modify/

Table 188: 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 189: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.13 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 190: 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 452) to retrieve the ID.

Table 191: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.14 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/modify/

Table 192: 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 452) to retrieve the ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.15 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/modify/

Table 193: 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 452) to retrieve the ID.
Email	Body	**Required**. A string indicating the updated email address of the recipient.

Table 194: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.16 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/modify/

Table 195: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.17 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 196: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.18 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/read/

Table 197: 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 198: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.19 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/modify/

Table 199: 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 200: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.20 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/modify/

Table 201: 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 456) to retrieve the ID.
Email	Body	**Required**. A string indicating the updated email address of the recipient.

Table 202: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.8.21 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_authorities/modify/

Table 203: 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: `https://keyfactor.keyexample.com/KeyfactorAPI/CertificateAuthority/Import?dns=keyother.com`

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9 Certificate Collections

The Certificate Collections component of the Keyfactor API is used to create, edit, list, and set permissions on certificate collections.

Table 204: 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 464
/	PUT	Updates an existing certificate collection.	PUT Certificate Collections on page 470
/{id}	GET	Returns the certificate collection with the specified ID.	GET Certificate Collections ID on page 474

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the certificate collection with the specified ID.	DELETE Certificate Collection ID on page 476
/{name}	GET	Returns the certificate collection with the specified name.	GET Certificate Collections Name on page 477
/Copy	POST	Creates a new certificate collection based on an existing collection.	POST Certificate Collections Copy on page 479
/NavItems	GET	Returns the list of *favorite* certificate collections (that have been set to *Show in Navigator*).	GET Certificate Collection Nav Items on page 485
/{id}/Favorite	PUT	Updates the *favorite* setting for the collection specified.	PUT Certificate Collection ID Favorite on page 485
/CollectionList	GET	Returns information about the definitions for all collections, including the de-duplication setting.	GET Certificate Collections List on page 486

2.6.9.1 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 205: 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* in the *Keyfactor Command Reference Guide*. 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 206: 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* in the *Keyfactor Command Reference Guide*. 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.2 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 207: 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 Pagein the Keyfactor Command Reference Guide 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 in the Keyfactor Command Reference Guide. 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 461) 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 208: 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* in the *Keyfactor Command Reference Guide*. Possible values are: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.3 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 in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 209: 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 461) 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*in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide*. The default is 0. Possible values are: Value: 0 — Description: None

<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>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>
<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 210: 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* in the *Keyfactor Command Reference Guide*. Possible values are: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.4 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 211: 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 461) to retrieve a list of all the certificate collections to determine the certificate collection ID.

Table 212: 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* in the *Keyfactor Command Reference Guide*. 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.5 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 213: 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 461) to retrieve a list of certificate collections to determine the collection ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.6 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 214: 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 305) 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 215: 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* in the *Keyfactor Command Reference Guide*. 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.7 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 216: 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 Pagein the Keyfactor Command Reference Guide 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 in the Keyfactor Command Reference Guide. 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 461) 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 217: 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* in the *Keyfactor Command Reference Guide*. Possible values are: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.8 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/collections/read/
OR
/certificates/collections/read/#/ (where # is a reference to a specific certificate collection ID)

Table 218: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.9 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 219: 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 461) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.9.10 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions.

Table 220: 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* in the *Keyfactor Command Reference Guide*. 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 221: 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* in the *Keyfactor Command Reference Guide*. 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10 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 222: Certificate Stores Endpoints.

Table 222: Certificate Stores Endpoints

Endpoint	Method	Description	
/	DELETE	Deletes multiple certificate stores specified in the request body.	DELETE Certificate Stores on page 491
/	GET	Returns all certificate stores with paging and option to specify detail level.	GET Certificate Stores on page 492
/	POST	Creates a new certificate store if valid parameters are supplied.	POST Certificate Stores on page 504
/	PUT	Updates an existing certificate store.	PUT Certificate Stores on page 532
/{id}	DELETE	Deletes a certificate store by its GUID.	DELETE Certificate Stores ID on page 562
/{id}	GET	Returns certificate store details for the specified certificate store.	GET Certificate Stores ID on page 563

Endpoint	Method	Description	
/{id}/Inventory	GET	Returns certificate inventory for the specified certificate store.	GET Certificate Stores ID Inventory on page 581
/Server (*deprecated)	GET	Returns a list of certificate store servers.	GET Certificate Stores Server on page 583
/Server (*deprecated)	POST	Creates a new certificate store server.	POST Certificate Stores Server on page 586
/Server (*deprecated)	PUT	Updates an existing certificate store server.	PUT Certificate Stores Server on page 591
/Password	PUT	Updates the password for a certificate store.	PUT Certificate Stores Password on page 596
/DiscoveryJob	PUT	Creates a job to find certificate stores.	PUT Certificate Stores Discovery Job on page 599
/AssignContainer	PUT	Assigns a certificate store to a container.	PUT Certificate Stores Assign Container on page 605
/Approve	POST	Approves an array of pending certificate stores.	POST Certificate Stores Approve on page 616
/Schedule	POST	Creates an inventory schedule for a certificate store.	POST Certificate Stores Schedule on page 627
/Reenrollment	POST	Schedules a reenrollment of a certificate into a certificate store.	POST Certificate Stores Reenrollment on page 630
/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 632

Endpoint	Method	Description	
/Certificates/Remove	POST	Configures a management job to remove a certificate from one or more stores with the provided schedule.	POST Certificate Stores Certificates Remove on page 639

2.6.10.1 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 223: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.2 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 224: 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* in the *Keyfactor Command Reference Guide*. 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 225: 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 643).
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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 713 for more

Name	Description

	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 1065):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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>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
<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>

<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>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>
</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**

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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.3 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 226: 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 643).
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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 713 for more information).

Name	In	Description

		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 1065):
		`"{ \"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 504).
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) in the Keyfactor Command Reference Guide and PAM Providers on page 1038 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

Name	In	Description
		Name: Description Name: Description Name: Description Name: Description Nam-e: Descrip-tion Para-ms: 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. Provider-Type Para-mValues: An array of objects containing the values for the provider types specified by Provider-TypeParams. See the previous level of *Provider-TypeParamValues* for details. Secured-AreaId: An integer indicating the Keyfactor Command refer-

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* in the *Keyfactor Command Reference Guide*. 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	In	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 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

Name	In	Description
		Name: Description Name: Description Name: Description provider (true). For an example, see GET PAM Providers on page 1065. 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

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>Descri-ption</th></tr><tr><td></td><td>the internal name for the PAM provide-r type para-meter.</td></tr><tr><td>Provider-TypeParams</td><td>Unused field</td></tr></table></td></tr></table></td></tr></table></td></tr><tr><td>Providerl-d</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider.</td></tr><tr><td>IsMan-aged</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 227: 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 643).
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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 713 for more

Name	Description

	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 1065):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
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>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
<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

Name	Description
	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: `"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
StoreTypeID	**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	**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.
DisplayName	**Required**. A string containing the full display name of the entry para-

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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.4 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 228: 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 643).
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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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	Bo-	**Required**. Some types of certificate stores have additional properties that are

Name	In	Description

	dy	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 713 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 1065):
		`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\":`

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>

<pre>{\"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.
</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>

<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>
</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>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:

<code>"JobProperties": ["sniCert", "virtualServerName"]</code>

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:

<code>"JobProperties": [{"sniCert":"MyCertificateName","virtualServerName":"MyVirtualServerName"}]</code>

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 504).
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) in the Keyfactor Command Reference Guide and PAM Providers on page 1038 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
		Name: Description Name: Description Name: Description Name: 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	In	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
		Name / Description Name / Description Name / Description *Keyfactor Command* in the *Keyfactor Command Reference Guide*. 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 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	In	Description
		Name / Description Name / Description Name / Description 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). For an example, see GET PAM Providers on page 1065. ProviderType: An object containing details for the provider type. Name / Description: Id — A string

Name	In	Description
		Name: Description Name: Description Name: Description Name: Description indicating 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 229: 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 643).
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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 713 for more

Name	Description

	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 1065):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`

Name	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	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

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>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>

<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>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>
</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.5 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 230: 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 492) to retrieve a list of all the certificate stores to determine the certificate store GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.6 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 231: 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 492) to retrieve a list of all the certificate stores to determine the certificate store GUID.

Table 232: 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 643).
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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 713 for more

Name	Description

	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 1065):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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>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 parameter.</td></tr></table></td></tr></table>

Name	Description
	Name / **Description** Name / Description (blank) — 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 field when configuring a remove certi-

Name	Description
	Name / **Description**

Name	Description
	ficate 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:

```
"EntryParameter": [
```

Name	Description

	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><pre>{
	"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 is using the certificate. In this case, the authorized</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>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>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 504).
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) in the Keyfactor Command Reference Guide and PAM Providers on page 1038 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:</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>"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 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</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>password resides).</td></tr><tr><td>Instance-Id</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>Instance-Guid</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><tr><td>Name</td><td>A string indicating the internal name for the PAM provider.</td></tr><tr><td>Area</td><td>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.</td></tr><tr><td>Provider-Type</td><td>An array of objects containing details about the provider type for the provider, including:</td></tr></table></td></tr></table></td></tr></table>

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* in the *Keyfactor Command Reference Guide*.</td></tr></table></td></tr><tr><td>Provider-</td><td>An array of objects that the provider type uses</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>Type Param</td><td>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 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></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>Instance-Level</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). For an example, see GET PAM Providers on page 1065.</td></tr><tr><td>Provider-Type</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></table></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>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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>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.
</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.7 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 233: 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 234: 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 in the Keyfactor Command Reference Guide 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 719 for more information about entry parameters.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.8 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 492).

Table 235: 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* in the *Keyfactor Command Reference Guide*. 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 236: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.9 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 504).

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 237: POST Certificate Stores Server Input Parameters

<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 1065) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use. See PAM Provider Configuration in Keyfactor Command 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 1065) 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 1065) 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 1065) 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 713) 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 238: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.10 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 532. 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 239: 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 <a>GET PAM Providers on page 1065) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use. See PAM Provider Configuration in Keyfactor Command 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 <a>GET PAM Providers on page 1065) 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 1065) 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 1065) 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 240: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.11 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 241: 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 492) 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 1065) 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 1065) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.12 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* in the *Keyfactor Orchestrators Installation and Configuration Guide*.
- 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 616).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 242: 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.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ServerUsername</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 <a>GET PAM Providers on page 1065) to retrieve a list of your defined PAM providers to determine the PAM provider ID to use. See PAM Provider Configuration in Keyfactor Command 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 <a>GET PAM Providers on page 1065) 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>"ServerUsername": {
  "Provider": 2,
  "Parameters": {
    "SecretId": 4
  }
},</pre>For CyberArk, this might be:
<pre>"ServerUsername": {
  "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>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 1065) 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 1065) 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>

Name	In	Description

		<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.

ServerUseSsl	Body	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*.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.13 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 243: 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 244: 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 643).
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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 713 for more

Name	Description

	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 1065):
	`"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyUserID\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"MyPasswordID\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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>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
<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>

<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>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>
</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.14 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 599). 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 245: 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 492) 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 643) 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 713 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 1065):
		<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) in the Keyfactor Command Reference Guide and PAM Providers on page 1038 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

<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></td><td>indicates the name of the provider type.</td></tr>
<tr><td>ProviderTypeParams</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 ProviderTypeParam for details.</td></tr>
</table>
</td></tr>
<tr><td>ProviderType</td><td>An array of objects containing the values for</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name: Description Name: Description Name: Description Name: Description ParamValues: 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 Keyfactor Command* in the *Keyfactor Command Reference Guide*. 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 1065.
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
		Name / Description Name / Description Name / Description Name / Description parameter. Name: A string indicating the internal name for the PAM provider type parameter. ProviderTypeParams: Unused field Provider: 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. This field is **required** for Java keystores.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.15 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 246: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.16 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 1011).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 in the *Keyfactor Command Reference Guide*) must have enrollment permissions configured on the CA and template.

Table 247: 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 492) 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 492) 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 in the *Keyfactor Command Reference Guide*).
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 in the *Keyfactor Command Reference Guide*).

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.17 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 1011).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 Permissions and *Container Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection and container permissions.

Table 248: POST Certificate Stores Certificates Add Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>CertificateId</td><td>Body</td><td>Required. An integer containing the Keyfactor Command reference ID of the certificate to be added to the certificate store(s).</td></tr>
<tr><td>Certi-ficateStores</td><td>Body</td><td>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:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Certi-ficateStoreIds</td><td>Required. A string containing the GUID for the certificate store to which the certificate should be added.</td></tr>
<tr><td>Alias</td><td>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 in the Keyfactor Command Reference Guide for more information. This field may be required depending on the store type selected.</td></tr>
<tr><td>JobFields</td><td>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.</td></tr>
<tr><td>Overwrite</td><td>A Boolean that sets whether a certificate in the store with the Alias provided should be over-written 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 279) to retrieve a list of the locations an</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>existing certificate is in to determine the alias 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 GET PAM Providers on page 1065) 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 GET PAM Providers on page 1065)</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>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>"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</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>certificate store. This is only relevant if there's a 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 Permissions in the Keyfactor Command Reference Guide for more information.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.10.18 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 1011).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 Permissions* and *Container Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection and container permissions.

Table 249: 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 287) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.11 Certificate Store Containers

The CertificateStoreContainers component of the Keyfactor API provides a set of methods to support management of certificate store containers.

Table 250: 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 645
/{id}	DELETE	Deletes a certificate store container.	DELETE Certificate Store Containers ID on page 675
/{id}	GET	Returns details for the specified	GET Certificate Store

Endpoint	Method	Description	Link
		certificate store container.	Containers ID on page 676
/{id}	PUT	Edits a certificate store container.	PUT Certificate Store Containers on page 650

2.6.11.1 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 251: 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* in the *Keyfactor Command Reference Guide*. 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 252: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.11.2 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 253: 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 254: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.11.3 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 255: PUT Certificate Store Containers 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 ID of the container.</td></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:</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>"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>CertStoreType</td><td>Body</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+. The default is 0 for a JKS keystore.</td></tr>
</table>

Table 256: 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 Certificate Store Operations: Adding or Modifying a Certificate Store in the Keyfactor Command Reference Guide 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</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description
	device (e.g. Common). See *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* for more information.
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 713 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\",`

Name	Description

Name	Description

```
   \"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\":\"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 1065):

```
"{
   \"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):

Name	Description

Name	Description
	• 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	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

Name	Description
	Name / **Description**
	cannot re-enroll, if applicable.

JobProperties	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 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

Name	Description

Name	Description

Name	Description
	on a certificate store of this type. Entry parameter options include:

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:

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Name / Description
	• **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 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 config-

Name	Description
	Name / **Description**
	Name / **Description**
	Name / **Description**
	Name / Description
	uring a reenrollment job.
	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

Name	Description

Name	Description

Name	Description

Name	Description
	multiple choice options for this entry parameter.

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.

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>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>SetNewPass-wordAllowed</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 Server on page 586). 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) in the Keyfactor Command Reference Guide and PAM Providers on page 1038 for more information. The possible values are:<table><tr><th>Name</th><th>Description</th></tr><tr><td>SecretV-</td><td>A string—submitted as an object—indicating a password</td></tr></table></td></tr></table>

Name	Description
	Name / **Description**

Name	Description
	Name / Description

Name	Description

alue	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:
	`"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 ParameterVa-	An array of objects containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:

Name	Description

Name	Description

Name	Description
lues	

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).
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	Description

Name	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.
ProviderType	An array of objects containing details about the provider type for the provider, including:

Name	Description
Id	A string indicating the Keyfactor Command

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	reference GUID for the provider type.
Name	A string that indicates the name of the provider type.
Provider Type Params	An array of parameters that the provider type uses for data input in Keyfactor Command

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><table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>when creating new PAM provider and certificate store records. See below instance of ProviderTypeParam 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></table></td></tr></table></td></tr></table></td></tr></table>

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
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* in the *Keyfactor Command Reference Guide*.

ProviderType Param	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

Name	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.
DataT-	An integer indicating the

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
ype	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 1065.
ProviderType	An object containing details for the provider type.

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
Id	A string indicating the Keyfactor Command reference GUID - for the PAM provider type parameter.
Name	A

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	string indicating the internal name for the PAM provider type parameter.
ProviderTypeParams	Unused field

Name	Description
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

Name	Description
	Name / **Description** **Name** / **Description** 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.11.4 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 257: 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 643) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.11.5 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 258: 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 643) to retrieve a list of all the certificate store containers to determine the certificate store container ID.

Table 259: 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* 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 713 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\":\"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 1065):
	
	`"{ \"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
	• 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 586). 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)* in the *Keyfactor Command Reference Guide* and PAM Providers on page 1038 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

Name	Description

Name	Description

Name	Description
	Keyfactor secrets table, include the password in quotes. For example: `"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

Name	Description

Name	Description

Name	Description

Name	Description
	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

Name	Description
	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 GUID for the provider type.
Name	A string

<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><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>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.</td></tr></table></td></tr></table></td></tr></table></td></tr></table></td></tr></table></td></tr></table>

Name	Description
	Name / **Description** Name / Description Name / Description Name / Description Name / Description Name / Description Na-me / Descrip-tion See below instance of *Provider-TypePara-m* for details. ProvideerType ParamValue-s / An array of objects containing the values for 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.

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	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* in the *Keyfactor Command Reference Guide*.

Name	Description
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

Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	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

Name	Description

Name	Description

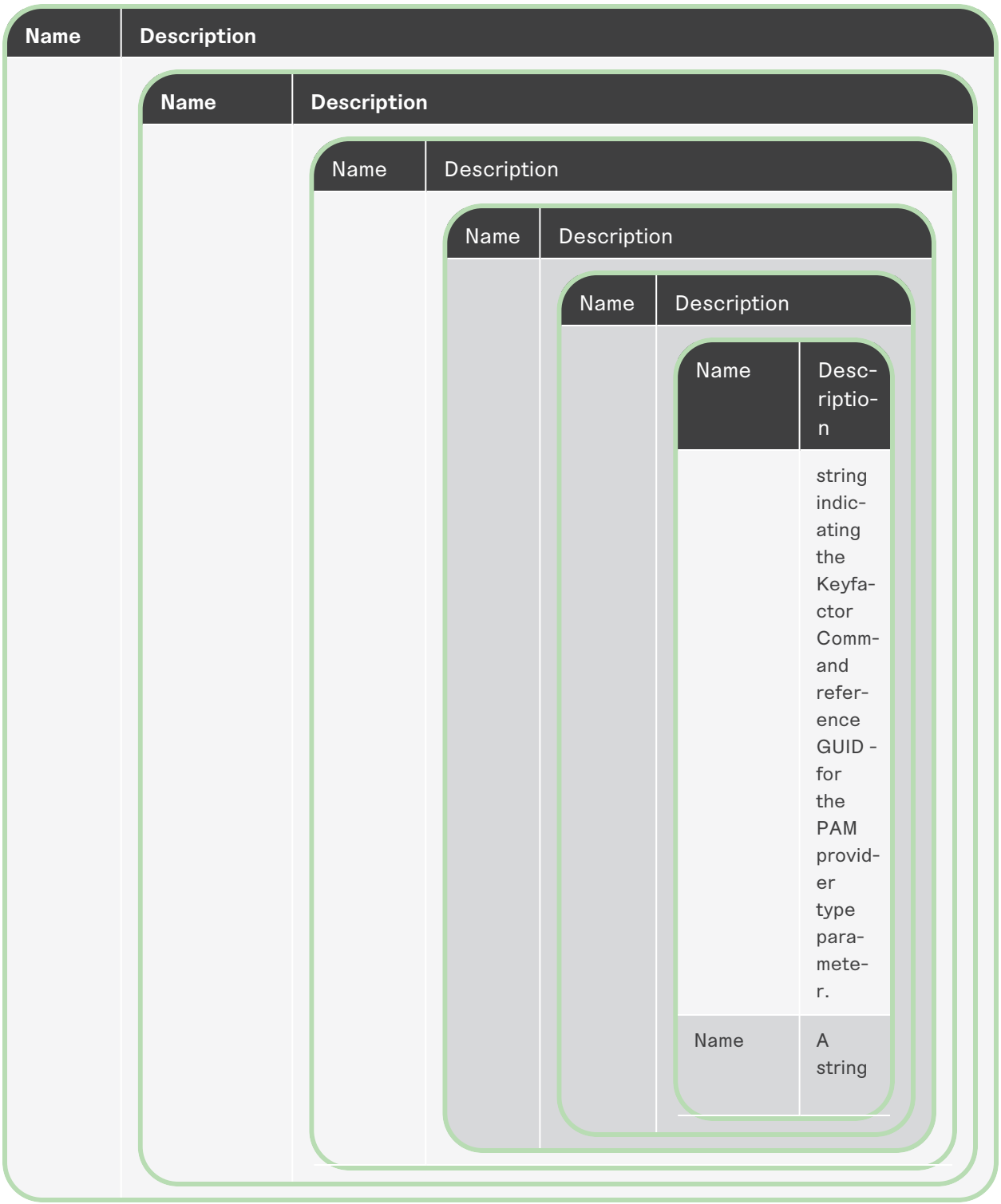
Name	Description

Name	Description

Name	Description
	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 1065.
ProviderType	An object containing details for the provider type.

Name	Description
Id	A

Name	Description



Name	Description

Name	Description

Name	Description

Name	Description

Name	Description
	string indicating the Keyfactor Command reference GUID - for the PAM provider type parameter.
Name	A string

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><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></td></tr></table>

Name	Description
	Name / **Description** **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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.12 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 260: Certificate Store Type Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes a certificate store type using StoreType number.	DELETE Certificate Store Types ID on the next page
/{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 705
/	DELETE	Delete multiple certificate store types using StoreType number.	DELETE Certificate Store Types on page 712
/	GET	Returns all certificate store types with paging and options to the specified detail level.	GET Certificate Store Types on page 713
/	POST	Creates a new certificate store type.	POST Certificate Store

Endpoint	Method	Description	Link
			Types on page 719
/	PUT	Updates a certificate store type using StoreType number.	PUT Certificate Store Types on page 733

2.6.12.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_stores/modify/

Table 261: 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 713) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.12.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_stores/read/ OR container permission

Table 262: 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 713) to retrieve a list of all the certificate store types to determine the certificate store type ID.

Table 263: 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 in theKeyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</td></tr>
</table>

Name	Description
CustomAliasAllowed	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).
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	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 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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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.</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

Name	Description
	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).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.12.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_stores/read/

Table 264: 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 713) to retrieve a list of all the certificate store types to determine the certificate store type short name.

Table 265: 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 in theKeyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</td></tr>
</table>

Name	Description
CustomAliasAllowed	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).
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	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 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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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.</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

Name	Description
	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).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.12.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_stores/modify/

Table 266: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.12.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_stores/read/ OR container permission

Table 267: 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 268: 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 in theKeyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</td></tr>
</table>

Name	Description
CustomAliasAllowed	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).
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	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 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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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.</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

Name	Description
	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).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.12.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_stores/modify/

Table 269: 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

Name	In	Description
		<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

<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 Application Settings: Enrollment Tab in the Keyfactor Command Reference Guide.

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>"StorePathValue": "
[\"Apple\",\"Cherry\",\"Peach\",\"Pear\"]"</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
</td></tr>
</table>

Name	In	Description
		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* in the *Keyfactor Command Reference Guide*. 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	In	Description
		<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><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*. </td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>• **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 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 270: 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 in theKeyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</td></tr>
</table>

Name	Description
CustomAliasAllowed	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).
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	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 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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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.</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

Name	Description
	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).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.12.7 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* in the *Keyfactor Command Reference Guide*) 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 271: 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 — **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 in the Keyfactor Command Reference Guide.
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* in the *Keyfactor Command Reference Guide*. 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 272: 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 in theKeyfactor Command Reference Guide.</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 in the Keyfactor Command Reference Guide.</td></tr>
</table>

Name	Description
CustomAliasAllowed	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).
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	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 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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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.</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

Name	Description
	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).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.13 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 273: 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

Endpoint	Method	Description	Link
/{id}	DELETE	Delete the Keyfactor Command server with the specified ID from the Keyfactor Command database.	DELETE Component Installation ID below

2.6.13.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/system_settings/modify/

Table 274: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.13.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/system_settings/read/

Table 275: 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* in the *Keyfactor Command Reference Guide*. 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 276: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.14 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 277: 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 755
/Generate	POST	Generate and configure a CSR request.	POST CSR Generation Generate on page 756

2.6.14.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/requests/manage/

Table 278: 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 755) to retrieve a list of all the pending CSRs to determine the CSR IDs.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon () at the top of the Management Portal page next to the **Log Out** button.

2.6.14.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/requests/manage/

Table 279: GET CSR Generation Pending {id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the CSR that should be retrieved.

Table 280: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.14.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/requests/manage/

Table 281: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.14.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/requests/manage/

Table 282: 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 283: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.14.5 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 825), 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 333).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 284: 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>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<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 in the Keyfactor Command Reference Guide 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</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>Abbreviation</th><th>Description</th></tr>
<tr><td></td><td></td><td>address of the certificate to be requested with the CSR.</td></tr>
</table>
</td></tr>
<tr><td>Template</td><td>Body</td><td>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.</td></tr>
</table>

Table 285: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.15 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 286: 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 765
/	POST	Creates a custom job type.	POST Custom Job Types on page 767
/	PUT	Updates an existing custom job type.	PUT Custom Job Types on page 771

2.6.15.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/modify/

Table 287: 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 765) to retrieve a list of all the custom job types to determine the job type GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.15.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 288: 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 765) to retrieve a list of all the custom job types to determine the job type GUID.

Table 289: GET JobTypes Custom {id} 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.15.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 290: 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 291: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.15.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/modify/

Table 292: 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 293: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.15.5 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* in the *Keyfactor Command Reference Guide*) 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 294: PUT JobTypes Custom Input Parameters

Name	In	Description
Id	Body	The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.
JobTypeName	Body	**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.
Description	Body	A string containing a description for the custom job type.
JobTypeFields	Body	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 — **Required**. A string that indicates the name for the job type field. Type — **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: Integer Value / **Enum Value** / Description 1 / String / String 2 / Int / Integer 3 / DateTime / Date 4 / Bool / Boolean DefaultValue — **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*. Required — A Boolean that sets whether the job type field is required (true) or not (false). The default is *false*.

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 295: PUT 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16 Enrollment

The Enrollment component of the Keyfactor API includes methods necessary to enroll certificate signing requests (CSRs) and personal information exchanges (PFXs).

Table 296: 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 783
/PFX/Context/My	GET	Returns the templates available for PFX enrollment by the current user.	GET Enrollment PFX Content My on page 801
/AvailableRenewal/Id/{id}	GET	Returns the type of renewals available for the referenced certificate ID.	GET Enrollment Available Renewal ID on page 820
/AvailableRenewal/Thumbprint/{thumbprint}	GET	Returns the type of renewals available for the referenced certificate thumbprint.	GET Enrollment Available Renewal Thumbprint on page 823
/CSR	POST	Performs a CSR enrollment.	POST Enrollment CSR on page 825
/PFX	POST	Performs a PFX enrollment.	POST Enrollment PFX on page 832
/CSR/Parse	POST	Returns information found in a CSR in a human friendly form.	POST Enrollment CSR Parse on page 850
/PFX/Deploy	POST	Adds a certificate into a certificate store following a PFX enrollment or certificate renewal.	POST Enrollment PFX Deploy on page 852
/PFX/Replace	POST	Replaces a certificate in a certificate store following a PFX enrollment.	POST Enrollment PFX Replace on page 858

Endpoint	Method	Description	Link
/Renew	POST	Performs a certificate renewal.	POST Enrollment Renew on page 861

2.6.16.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/enrollment/csr/
OR
/certificates/enrollment/pfx/
OR
/certificates/enrollment/csr/generation/

Table 297: 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 1593) to retrieve a list of all the templates to determine the template ID.

Table 298: GET Enrollment Settings {id} Response Data

Name	Description
TemplateRegexes	An array of objects containing the regular expressions resolved for the template. 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 1592) 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 character in the Common Name field in the enrollment pages.

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
	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$`
	UPN (Subject Alternative Name: User Principal Name) — This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
	Error — A string specifying the error message displayed to the user when

<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 1566. 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 1592) 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>

<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>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.</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 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: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.2 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 8). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/enrollment/csr/

Table 299: 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	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* in the *Keyfactor Command Reference Guide* for more information.

Name	Description
	Name / **Description**
	EnrollmentFields — 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 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 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>

<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>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 user enters information in a metadata field that does not match the template-specific regular expression (Validation 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>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 1592) 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**
	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 1566. 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 in the Keyfactor Command Reference Guide for more information.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.3 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 8). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/enrollment/pfx/

Table 300: 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* in the *Keyfactor Command Reference Guide* for more information.

Name	Description
	Name / **Description**
	EnrollmentFields — 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 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 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>

<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>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>
</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\\.org|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-Business,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 1592) 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**
	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 1566. 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 in the Keyfactor Command Reference Guide for more information.</td></tr></table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.4 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the note under the *CollectionId* input parameter, below.

Table 301: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Table 302: 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 in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.5 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection permissions. See also the note under the *CollectionId* input parameter, below.

Table 303: 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 Permissions* in the *Keyfactor Command Reference Guide* for more information.

Table 304: 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 Certificate Templates in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.6 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*).

Table 305: 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 *host-name**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 Certificate Metadata in the Keyfactor Command Reference Guide 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 <a>GET Templates on page 1593) 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 in the Keyfactor Command Server Installation Guide) 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 Configuring Template Options in the Keyfactor</td></tr>
</table>

Name	In	Description
		Command Reference Guide for more information.
x-CertificateFormat	Header	**Required**. A string indicating the desired output format for the certificate. Available options are DER and PEM.

Table 306: POST Enrollment CSR Response Data

Value	Description
CertificateInformation	An object containing information about the certificate that was requested. CSR 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.
Certificates	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*.
WorkflowInstanceId	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.
WorkflowReferenceId	An integer containing the Keyfactor Command reference ID of the workflow instance.

<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 in the Keyfactor Command Reference Guide for more information.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.7 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* in the *Keyfactor Command Reference Guide*) 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 846 parameter) using the POST /Enrollment/PFX/Deploy method (see POST Enrollment PFX Deploy on page 852) or POST /Enrollment/PFX/Replace method (see POST Enrollment PFX Replace on page 858).

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* in the *Keyfactor Command Reference Guide*).

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 11.

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 307: 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 *Configuring Template Options* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide* for more information.

Name	In	Description
IncludeChain	Body	A Boolean that sets whether to include the certificate chain in 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`

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>

<pre>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 *Certificate Metadata* in the *Keyfactor Command Reference Guide* for more information.
</td></tr>
<tr><td>Password</td><td>Body</td><td>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 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>RenewalCertificateId</td><td>Body</td><td>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.</td></tr>
<tr><td>Stores</td><td>Body</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 GUID of the certificate store to which the certificate should be deployed.
Use the GET /CertificateStores method (see <a>GET Certificate Stores on page 492) 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>
</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>Alias</td><td>A string containing 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 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 <a>GET Certificates Locations ID on page 279) 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</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>Job Custom Fields.
The setting is referenced using the following format:
<pre>"Properties":
{"NetscalerVserver":"MyVirtualSe
rverName"}</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>Subject</td><td>Body</td><td>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 in the Keyfactor Command Reference Guide 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</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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><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:</td></tr>
</table>

Name	In	Description
		• JKS Select the JKS option when you intend to create a Java 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 858). 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 852). 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 308: 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 <a>POST Enrollment PFX Deploy on page 852).</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 Certificate Metadata in the Keyfactor Command Reference Guide 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 309: 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 *Configuring Template Options* in the *Keyfactor Command Reference Guide* 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* in the *Keyfactor Command Reference Guide* 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:

Name	In	Description
		• 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
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": "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. }` See *Certificate Metadata* in the *Keyfactor Command Reference Guide* for more information.
Password	Body	**Required**. A string that sets the password used to encrypt the contents of the PFX file. The minimum pass-

Name	In	Description
		word length is controlled by the *Password Length* application setting. The default is *12*. See *Application Settings: Enrollment Tab* in the *Keyfactor Command Reference Guide* 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": {`

Name	In	Description

		<pre>"dns": [
		"dnssan1.keyexample.com",
		"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* in the *Keyfactor Command Reference Guide* 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:

Name	In	Description
		• JKS Select the JKS option when you intend to create a Java 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 858). 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 852). 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 310: 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 852).</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 in the Keyfactor Command Reference Guide for more information.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.8 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
None

Table 311: 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 312: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.9 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 832) 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 313: 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 <a>GET Certificate Stores on page 492) 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 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 <a>GET Certificates Locations ID on page 279) 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:
<code>"JobProperties": ["NetscalerVserver"]</code>

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:
<code>"Properties": {"NetscalerVserver":"MyVirtualServerName"}</code>

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 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 279) 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 314: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.10 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 832) 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 852).

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 315: 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 316: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.16.11 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* in the *Keyfactor Command Reference Guide*) 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 Permissions* in the *Keyfactor Command Reference Guide* 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 852) or POST /Enrollment/PFX/Replace method (see POST Enrollment PFX Replace on page 858).

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* in the *Keyfactor Command Reference Guide*).

Table 317: 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 820 or GET Enrollment Available Renewal Thumbprint on page 823).
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 820 or GET Enrollment Available Renewal Thumbprint on page 823).

Table 318: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.17 Event Handler Registration

EventHandlerRegistration endpoints includes methods necessary to list, add, edit and delete event handler registrations for Alerts in Keyfactor Command.

Table 319: EventHandlerRegistration Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes an event handler.	DELETE Event Handler Registration ID on page 870
/{id}	GET	Returns a registered event handler that matches the provided ID.	GET Event Handler Registration ID on page 867
/{id}	PUT	Updates a registered event handler's information.	PUT Event Handler Registration ID on page 869
/	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 866

2.6.17.1 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* in the *Keyfactor Command Reference Guide*). 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/handlers/registration/read/

Table 320: 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*. 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 321: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.17.2 POST Event Handler Registration

The POST /Event Handler Registration method is used to register an event handler (see *Event Handler Registration* in the *Keyfactor Command Reference Guide*). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/handlers/registration/modify/

Table 322: 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* in the *Keyfactor Command Reference Guide*).

Table 323: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.17.3 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* in the *Keyfactor Command Reference Guide*). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/handlers/registration/read/

Table 324: 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 864) to retrieve a list of all the event handlers to determine the ID.

Table 325: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.17.4 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* in the *Keyfactor Command Reference Guide*). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/handlers/registration/modify/

Table 326: 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 864) 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 327: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.17.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/handlers/registration/modify/

Table 328: 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 864) to retrieve a list of all the event handlers to determine the ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.18 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 in the *Keyfactor Command Reference Guide* 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 329: 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 873
/	POST	Adds a new script.	POST Extensions Scripts on page 875
/	PUT	Updates a script.	PUT Extensions Scripts on page 879

2.6.18.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/scripts/modify/

Table 330: 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 873) to retrieve a list of all the scripts to determine the ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.18.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/scripts/read/

Table 331: 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 332: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.18.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/scripts/read/

Table 333: 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*. 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 334: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.18.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/scripts/modify/

Table 335: 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 336: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.18.5 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* in the *Keyfactor Command Reference Guide*) 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 337: 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 873) 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 875 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 338: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.19 Identity Providers

The Identity Providers component of the Keyfactor API includes methods necessary to programmatically edit and retrieve identity providers within Keyfactor Command.

Table 339: Identity Providers Endpoint

Endpoint	Method	Description	Link
/{id}	GET	Returns the identity provider with the specified	GET Identity

Endpoint	Method	Description	Link
		ID.	Providers ID on the next page
/{id}	PUT	Updates the identity provider with the specified ID.	PUT Identity Providers ID on page 896
/	GET	Returns all identity providers defined within Keyfactor Command with filtering and output options.	GET Identity Providers on page 926
/Types	GET	Returns details for all the identities provider types defined within Keyfactor Command.	GET Identity Providers Types on page 941

2.6.19.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/identity_providers/read/

Table 340: 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 926) to retrieve a list of all the identity providers to determine the provider's ID.

Table 341: 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 342: Identity Provider Parameters) contains the data shown in Table 343: Identity Provider Parameter Structure. Each parameter (Table 342: Identity Provider Parameters) contains the data shown in Table 343: Identity Provider Parameter Structure.

Table 342: 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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). 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*).

Name	Description

Name	Type	Example	Description
			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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). 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 Config-

Name	Description

Name	Type	Example	Description
			uration page, a link to which can be found on the Realm Settings page (see *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 docu-

Name	Description
	<table><tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr><tr><td></td><td></td><td></td><td>  ment 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.</td></tr><tr><td>Authorization Endpoint</td><td>1 - String</td><td>https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-</td><td>The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider,</td></tr></table>

Name	Description

Name	Type	Example	Description
		connect/auth	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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 *Gathering Keyfactor Identity Provider Data for the*

<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>
Keyfactor Command Installation in the Keyfactor Command Server Installation Guide 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) in the Keyfactor Command Reference Guide 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</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>PAM is not used as the storage location.</td></tr>
<tr><td>Parameters</td><td>An object 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 1065) to retrieve a list of all the PAM providers to determine the ID.</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>
For example, a username stored as a Keyfactor secret will look like:
<pre>{
   "SecretValue":
"KEYEXAMPLE\svc_MySer-
viceName"
}</pre>
For example, a password stored as a Keyfactor secret will look like:
<pre>{
   "SecretValue": "MySu-
perSecretPassword"
}</pre>
A secret stored as a CyberArk PAM secret will look like (where the Provider value—1 in this example—is the Id value from <a>GET PAM Providers on page 1065 and the Safe, Folder, and Object reference the information in the CyberArk safe needed for this record):
<pre>{
   "Provider": "1",
   "Parameters":{
      "Safe":"MySafeName",
      "Folder":"MyFolder-
Name",
      "Object":"MyOb-
jectName"
   }
}</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>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>A secret stored as a Delinea PAM secret will look like (where the Provider value—2 in this example—is the Id value from <a>GET PAM Providers on page 1065 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 Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation in the Keyfactor Command Server Installation Guide). 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</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description
			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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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:

Name	Description

Name	Type	Example	Description
			`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.
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

Name	Description

Name	Type	Example	Description
			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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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):

Name	Description

Name	Type	Example	Description
			`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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 343: Identity Provider Parameter Structure

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the parameter.</td></tr><tr><td>Name</td><td>A string indicating the short reference name for the parameter (e.g. NameClaimType).</td></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.19.2 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* in the *Keyfactor Command Reference Guide*) 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 344: 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 345: Identity Provider Parameters) contains the data shown in Table 346: Identity Provider Parameter Structure. Each parameter (Table 345: Identity Provider Parameters) contains the data shown in Table 346: Identity Provider Parameter Structure. *Table 345: 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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). Keyfactor recommends that you use a different client for this purpose than the client used

Name	In	Description

Name	Type	Example	Description
			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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). 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.
Authorit-	1 -	https://my-keyidp-	The issuer/authority endpoint

Name	In	Description

Name	Type	Example	Description
y	String	server.keyexample.com/realms/Keyfactor	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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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

Name	In	Description

Name	Type	Example	Description
			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 must match what's in the discovery

Name	In	Description

Name	Type	Example	Description
			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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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:

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><code>Command-OIDC-Client</code> For more information, see *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). This parameter is required.</td></tr><tr><td>Client Secret</td><td>2 - Secret</td><td></td><td>The secret for the client application created in the identity provider for primary application use. For Keyfactor Identity Provider, see *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide* 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</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>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>Command database.
• Load the secret information from a PAM provider.
See Privileged Access Management (PAM) in the Keyfactor Command Reference Guide 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 indicating the parameters to supply for PAM authentication. These will vary depending on the PAM provider.</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>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>Provider</td><td>A string indicating the ID of the PAM provider.
Use the GET /PamProviders method (see <a>GET PAM Providers on page 1065) 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:
<pre>{
   "SecretValue":
"KEYEXAMPLE\svc_MySer-
viceName"
}</pre>
For example, a password stored as a Keyfactor secret
</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>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>will look like:

<pre>{
   "SecretValue": "MySuperSecretPassword"
}</pre>
A secret stored as a CyberArk PAM secret will look like (where the Provider value—1 in this example—is the Id value from <a>GET PAM Providers on page 1065 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":"MyObjectName"
   }
}</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 <a>GET PAM Providers on page 1065 and the SecretId and SecretFieldName contain the information created in the</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>Type</th><th>Example</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>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 Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation in the Keyfactor Command Server Installation Guide). 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 data-</td></tr>
</table>
</td></tr>
</table>

Name	In	Description

Name	Type	Example	Description
			base.
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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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`

Name	In	Description

Name	Type	Example	Description
			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.
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

Name	In	Description

Name	Type	Example	Description
			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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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.

Name	In	Description

Name	Type	Example	Description
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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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:

Name	In	Description

Name	Type	Example	Description
			`https://<host>/admin/realms/<realm_name>` This parameter is required.

Table 346: 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).
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 347: 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 345: Identity Provider Parameters) contains the data shown in Table 346: Identity Provider Parameter Structure. Each parameter (Table 345: Identity Provider Parameters) contains the data shown in Table 346: Identity Provider Parameter Structure.

Table 348: 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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). 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*).

Name	Description

Name	Type	Example	Description
			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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). 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 Config-

Name	Description

Name	Type	Example	Description
			uration page, a link to which can be found on the Realm Settings page (see *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 docu-

Name	Description
	<table><tr><th>Name</th><th>Type</th><th>Example</th><th>Description</th></tr><tr><td></td><td></td><td></td><td>ment 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.</td></tr><tr><td>Authorization Endpoint</td><td>1 - String</td><td>https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-</td><td>The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider,</td></tr></table>

Name	Description

Name	Type	Example	Description
		connect/auth	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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 *Gathering Keyfactor Identity Provider Data for the*

<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>
Keyfactor Command Installation in the Keyfactor Command Server Installation Guide 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) in the Keyfactor Command Reference Guide 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</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>PAM is not used as the storage location.</td></tr>
<tr><td>Parameters</td><td>An object 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 1065) to retrieve a list of all the PAM providers to determine the ID.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description

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 look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1065 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"
   }
}
```

<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>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 1065 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 Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation in the Keyfactor Command Server Installation Guide). 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</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description
			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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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:

Name	Description

Name	Type	Example	Description
			`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.
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

Name	Description

Name	Type	Example	Description
			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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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):

Name	Description

Name	Type	Example	Description
			`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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 349: Identity Provider Parameter Structure

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the parameter.</td></tr><tr><td>Name</td><td>A string indicating the short reference name for the parameter (e.g. NameClaimType).</td></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.19.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
SecuritySettings: *Read*

Table 350: 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* in the *Keyfactor Command Reference Guide*. 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 351: GET Identity Provider 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 352: Identity Provider Parameters) contains the data shown in Table 353: Identity Provider Parameter Structure. Each parameter (Table 352: Identity Provider Parameters) contains the data shown in Table 353: Identity Provider Parameter Structure.

Table 352: 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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). 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*).

Name	Description

Name	Type	Example	Description
			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 *Using Keyfactor Identity Provider: Service Accounts* in the *Keyfactor Command Server Installation Guide*). 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 Config-

Name	Description

Name	Type	Example	Description
			uration page, a link to which can be found on the Realm Settings page (see *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 docu-

Name	Description

Name	Type	Example	Description
			ment 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-	The authorization endpoint URL for the identity provider. For Keyfactor Identity Provider,

Name	Description

Name	Type	Example	Description
		connect/auth	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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 *Gathering Keyfactor Identity Provider Data for the*

Name	Description

Name	Type	Example	Description
			Keyfactor Command Installation in the *Keyfactor Command Server Installation Guide* 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)* in the *Keyfactor Command Reference Guide* for more information.

Value	Description
SecretValue	A string containing the secret. This parameter is used when

<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>PAM is not used as the storage location.</td></tr>
<tr><td>Parameters</td><td>An object 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 1065) to retrieve a list of all the PAM providers to determine the ID.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description

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 look like (where the Provider value—1 in this example—is the Id value from GET PAM Providers on page 1065 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"
   }
}
```

<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>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 1065 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 Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation in the Keyfactor Command Server Installation Guide). 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</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Type	Example	Description
			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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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:

Name	Description

Name	Type	Example	Description
			`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.
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

Name	Description

Name	Type	Example	Description
			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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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):

Name	Description

Name	Type	Example	Description
			`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 *Gathering Keyfactor Identity Provider Data for the Keyfactor Command Installation* in the *Keyfactor Command Server Installation Guide*). 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 353: Identity Provider Parameter Structure

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the parameter.</td></tr><tr><td>Name</td><td>A string indicating the short reference name for the parameter (e.g. NameClaimType).</td></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.19.4 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/identity_providers/read/

Table 354: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.20 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 355: License Endpoint

Endpoint	Method	Description	Link
/	GET	Returns the current license.	GET License below

2.6.20.1 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 8). For more information regarding licensing, see *Licensing* in the *Keyfactor Command Reference Guide*.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
SystemSettings > Read

Table 356: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.21 MacEnrollment

The MacEnrollment component of the Keyfactor API includes methods to edit and retrieve the configuration for Mac auto-enrollment.

Table 357: 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

2.6.21.1 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/mac/auto-enrollment/management/read/

Table 358: 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* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.21.2 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* in the *Keyfactor Command Reference Guide*) 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 359: 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 360: 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* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.22 MetadataFields

MetadataFields contains definitions for metadata that can be associated with certificates in Keyfactor Command.

Table 361: MetadataFields Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes an existing metadata field.	DELETE Metadata Fields ID on the next page

Endpoint	Method	Description	Link
/{id}	GET	Returns detailed information for the specified metadata field.	GET Metadata Fields ID on the next page
/{name}	GET	Returns detailed information for the specified metadata field.	GET Metadata Fields Name on page 954
/{id}/InUse	GET	Returns a Boolean stating whether the metadata type is associated with a certificate.	GET Metadata Fields ID In Use on page 958
/	DELETE	Deletes multiple metadata fields specified in the request body.	DELETE Metadata Fields on page 959
/	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 960
/	POST	Creates a new metadata field using values supplied in the request body.	POST Metadata Fields on page 964
/	PUT	Updates an existing metadata field using values supplied in the request body.	PUT Metadata Fields on page 970

2.6.22.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/metadata/types/modify/

Table 362: 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 960) 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 958) to determine whether a metadata field is in use.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.22.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/metadata/types/read/

Table 363: 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 960) to retrieve a list of all the metadata fields to determine the metadata field's ID.

Table 364: 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 1548).</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 1548).</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 1548).</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 1548).
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 1548).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.22.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/metadata/types/read/

Table 365: 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 366: 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 GET Templates ID on page 1548).</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 1548).</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 1548).</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 1548).
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 1548).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.22.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:

/metadata/types/read/

Table 367: 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 368: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.22.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/metadata/types/modify/

Table 369: 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 958) to determine whether a metadata field is in use.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.22.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/metadata/types/read/

Table 370: 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* in the *Keyfactor Command Reference Guide*. 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 371: 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 <a>GET Templates ID on page 1548).</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 1548).</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 1548).</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 1548).
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 1548).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.22.7 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/metadata/types/modify/

Table 372: 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 1548).</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 1548).</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 1548).
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 1548).
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 1548).
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 373: 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 <a>GET Templates ID on page 1548).</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 1548).</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 1548).</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 1548).
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 1548).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.22.8 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* in the *Keyfactor Command Reference Guide*) 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 374: 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 1548).</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 1548).</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 1548).
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 1548).
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 1548).
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 375: 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 1548).</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 1548).</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 1548).</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 1548).
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 1548).
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.23 Monitoring

The Monitoring component of the Keyfactor API provides a set of methods to support management of CRL and OCSP monitoring locations.

Table 376: Monitoring Endpoints

Endpoint	Method	Description	Link
/Revocation/	POST	Creates a new revocation monitoring location.	POST Monitoring Revocation on page 988
/Revocation/	PUT	Edits the revocation monitoring location with the specified ID.	PUT Monitoring Revocation on page 996
/Revocation/	GET	Returns details for all revocation monitoring location according to the provided filter and output parameters.	GET Monitoring Revocation on page 983
/ResolveOSCP	POST	Resolves the given OCSP certificate authority.	POST Monitoring Resolve OSCP on page 1004
/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 1005
/Revocation/TestAll	POST	Tests the revocation monitoring alerts.	POST Monitoring Revocation Test All on page 1007

2.6.23.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 377: 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 983) to retrieve a list of all the revocation monitoring locations to determine the ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.23.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 378: 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 983) to retrieve a list of all the revocation monitoring locations to determine the ID.

Table 379: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.23.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 380: 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* in the *Keyfactor Command Reference Guide*. 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 381: GET Monitoring Revocation 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.23.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 382: 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 365) 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 1004) 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 1004) 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 1004) 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 1004) 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 383: 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:
<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 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.23.5 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* in the *Keyfactor Command Reference Guide*) 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 384: 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>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 GET Certificate Authority on page 365) 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 1004) 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 1004) 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 1004) 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 1004) 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 385: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.23.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 386: 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 365) to retrieve a list of all the CAs to determine the ID. One of either *CertificateContents* or *CertificateAuthorityId* is required, but not both.

Table 387: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.23.7 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

/monitoring/alerts/test/

Table 388: 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 389: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.23.8 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/
/monitoring/alerts/test/

Table 390: 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 391: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.24 Orchestrator Jobs

The Orchestrator Jobs component of the Keyfactor API includes methods necessary to schedule orchestrator jobs and view the results of jobs.

Table 392: 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 1011

Endpoint	Method	Description	Link
/ScheduledJobs	GET	Returns the details of active scheduled jobs, including in-process jobs.	GET Orchestrator Jobs Scheduled Jobs on page 1017
/Custom	POST	Schedules a custom job on the orchestrator using the provided information.	POST Orchestrator Jobs Custom on page 1021
/Reschedule	POST	Reschedules a failed orchestrator job.	POST Orchestrator Jobs Reschedule on page 1026
/Unschedule	POST	Unschedules an active orchestrator job.	POST Orchestrator Jobs Unschedule on page 1028
/Acknowledge	POST	Sets the status of a failed orchestrator job to acknowledged.	POST Orchestrator Jobs Acknowledge on page 1030
/Custom/Bulk	POST	Schedules a custom job on multiple orchestrator using the provided information.	POST Orchestrator Jobs Reschedule on page 1026

2.6.24.1 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* in the *Keyfactor Command Reference Guide*) 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 *Orchestrator Management Operations: Fetch Logs* in the *Keyfactor Command Reference Guide* for more information.

Table 393: 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 on the next page) to retrieve a list of jobs to determine the job's history ID.

Table 394: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.24.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 395: 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* in the *Keyfactor Command Reference Guide*. The query fields supported for this endpoint are: • AgentId (The GUID of the orchestrator. Run GET Agents on page 17 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 396: 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>"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>
<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).
OperationStart	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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* for more information.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.24.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/read/

Table 397: 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* in the *Keyfactor Command Reference Guide*. The query fields supported for this endpoint are: • AgentId (The GUID of the orchestrator. Run GET Agents on page 17 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 17) • 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 398: 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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 *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.24.4 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* in the *Keyfactor Command Reference Guide*) 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 1011) 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 1009) to retrieve the data.

Table 399: 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 17) 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 17) 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 765) 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 765) 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 400: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.24.5 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 627).

- 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 1021).
- 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 1021).
- SSL Discovery and Monitoring
  Change the schedule on these using PUT /SSL/Networks (see PUT SSL Networks on page 1518).
- 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 413).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 401: 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 1011) 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 *Orchestrator Scheduled Job Search Feature* in the *Keyfactor Command Reference Guide*. Either a list of one or more *JobAuditIds* or a *Query* is **required**, but not both.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.24.6 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 627).
- SSH Synchronization
  Change the schedule on these using PUT /SSH/ServerGroups (see PUT SSH Server Groups on page 1380).
- SSL Discovery and Monitoring
  Change the schedule on these using PUT /SSL/Networks (see PUT SSL Networks on page 1518).
- 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 413).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs container permissions.

Table 402: 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 1017) 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 *Orchestrator Scheduled Job Search Feature* in the *Keyfactor Command Reference Guide*. Either a list of one or more *JobIds* or a *Query* is **required**, but not both.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.24.7 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* in the *Keyfactor Command Reference Guide*). This endpoint returns 204 with no content upon success.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/agents/management/modify/

Table 403: 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 1011) 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 *Orchestrator Scheduled Job Search Feature* in the *Keyfactor Command Reference Guide*. Either a list of one or more *JobAuditIds* or a *Query* is **required**, but not both.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.24.8 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* in the *Keyfactor Command Reference Guide*) 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 1011) 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 1009) to retrieve the data.

Table 404: POST Orchestrator Jobs Custom Bulk Input Parameters

Name	In	Description
OrchestratorIds	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 17) 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 17) 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.
JobTypeName	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 765) 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 765) 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 405: 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: JobId — A string indicating the Keyfactor Command reference GUID for the job. Value: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.25 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 406: PamProviders Endpoints

Endpoint	Method	Description	Link
/	GET	Returns a list of all the configured PAM providers.	GET PAM Providers on page 1065
/	POST	Creates a new PAM provider.	POST PAM Providers on page 1083
/	PUT	Updates a PAM provider.	PUT PAM Providers on page 1102
/{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 1055
/Types	POST	Creates a new PAM provider type.	POST PAM Providers Types on page 1058
/Types	GET	Returns the PAM provider type with the specified ID.	GET PAM Providers Types ID on page 1123

2.6.25.1 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 11.

Table 407: 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 1065) to retrieve a list of all the PAM providers to determine the PAM provider's ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.25.2 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 11.

Version 2

Version 2 of the GET /PamProviders/{id} method has been redesigned to remove references to PAM associations with areas and containers.

Table 408: 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 1065) to retrieve a list of all the PAM providers to determine the provider's ID.

Table 409: 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
	<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/{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 410: 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 1065) to retrieve a list of all the PAM providers to determine the provider's ID.

Table 411: 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.

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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.25.3 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 11. The data returned in the response is the same for both versions.

Table 412: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.25.4 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 11. The input parameters and response data returned are the same for both versions.

Table 413: 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

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 414: POST 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.25.5 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 11.

Version 2

Version 2 of the GET /PamProviders method has been redesigned to remove references to PAM associations with areas and containers.

Table 415: 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* in the *Keyfactor Command Reference Guide*. 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 416: 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 417: 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* in the *Keyfactor Command Reference Guide*. 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 418: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.25.6 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 11.

Version 2

Version 2 of the POST /PamProviders method has been redesigned to remove references to PAM associations with areas and containers.

Table 419: 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 Installing Custom PAM Provider Extensions in the Keyfactor Command Reference Guide).</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</td></tr></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>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 provider type parameter. For parameters</td></tr></table></td></tr></table>

Name	In	Description
		Value / Description
		Value / Description
		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 certificate store to use the PAM provider

<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>(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 420: 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 421: 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 Installing Custom PAM Provider Extensions in the Keyfactor Command Reference Guide).</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</td></tr></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>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 provider type parameter. For parameters</td></tr></table></td></tr></table>

Name	In	Description
		Value / Description
		Value / Description
		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 certificate store to use the PAM provider

<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>(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 422: 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. 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://websrvr38.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>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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.25.7 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* 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 11.

Version 2 of the PUT /PamProviders method has been redesigned to remove references to PAM associations with areas and containers.

Table 423: 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 Installing Custom PAM Provider Extensions in the Keyfactor Command Reference Guide).</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.</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>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>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</td></tr></table></td></tr></table>

Name	In	Description
		Value / Description Value / Description 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. 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

<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>configuring a 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</td></tr>
</table>

Name	In	Description
		Keyfactor Command (false) or local to the orchestrator (true). The default is *false*.

Table 424: 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 425: 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 in the Keyfactor Command Reference Guide).</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.</td></tr></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>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>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</td></tr></table></td></tr></table>

Name	In	Description
		Value / Description Value / Description 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. 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

<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>configuring a 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</td></tr>
</table>

Name	In	Description
		Keyfactor Command (false) or local to the orchestrator (true). The default is *false*.

Table 426: 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. 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
	Value / Description

Value	Description
	Value / Description

Value	Description
	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: • 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.

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-

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><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>
ProviderTypeParamValues	An array of objects containing the values for the provider types specified by ProviderTypeParam. The field does not return data on responses.
Remote	A Boolean indicating whether the PAM provider is local to the Keyfactor Command (false) or local to the orchestrator (true). The default is *false*.
SecureAreaId	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.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.25.8 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide* for more information about global vs PAM provider permissions.

Table 427: 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 1065) to retrieve a list of all the PAM providers to determine the PAM provider's ID.

Table 428: GET PamProviders Types {id} 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.26 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 429: 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

2.6.26.1 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 430: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.27 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 1250) 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 in the *Keyfactor Command Reference Guide*) 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 431: 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 1132
/	POST	Adds a new permission set into Keyfactor Command.	POST Permission Sets on page 1133
/	PUT	Updates a permission set.	PUT Permission Sets on page 1134

2.6.27.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 432: 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 1132) to retrieve a list of all the permission sets to determine the permission set's ID.

Table 433: 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 *Security Role Operations: Version Two Permission Model* in the *Keyfactor Command Reference Guide* for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.27.2 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 434: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.27.3 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 1128). This method returns HTTP 200 OK on a success with details for the permission sets.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 435: 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 436: 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 *Security Role Operations: Version Two Permission Model* in the *Keyfactor Command Reference Guide* for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.27.4 POST Permission Sets

The POST /PermissionSets method is used to create a new permission set in Keyfactor Command (see Permission Sets on page 1128). 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 437: 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 *Security Role Operations: Version Two Permission Model* in the *Keyfactor Command Reference Guide* 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 438: 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 *Security Role Operations: Version Two Permission Model* in the *Keyfactor Command Reference Guide* for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.27.5 PUT Permission Sets

The PUT /PermissionSets method is used to update a permission set in Keyfactor Command (see Permission Sets on page 1128). 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 439: 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 *Security Role Operations: Version Two Permission Model* in the *Keyfactor Command Reference Guide* 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 440: 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 *Security Role Operations: Version Two Permission Model* in the *Keyfactor Command Reference Guide* for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28 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 441: 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 1145
/Custom/{id}	GET	Returns the custom report with the specified ID.	GET Reports Custom ID on page 1146
/Schedules/{id}	DELETE	Deletes the schedule for the built-in report with the specified schedule ID.	DELETE Reports Schedules ID on page 1147
/Schedules/{id}	GET	Returns the schedule for the built-in report with the specified schedule ID.	GET Reports Schedules ID on page 1148
/{id}/Parameters	GET	Returns the parameters for the built-in report with the specified report ID.	GET Reports ID Parameters on page 1152
/{id}/Parameters	PUT	Updates the parameters for the built-in report with the specified report ID.	PUT Reports ID Parameters on page 1155
/	GET	Returns all built-in reports with filtering and output options.	GET Reports on page 1157
/	PUT	Updates the built-in report with the specified ID. Only some fields can be updated.	PUT Reports on page 1160
/Custom	GET	Returns all custom reports with filtering and output options.	GET Reports Custom on page 1163
/Custom	POST	Creates a custom report.	POST Reports Custom on page 1165
/Custom	PUT	Updates the custom report with the specified ID.	PUT Reports Custom on page 1167
/{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 1168
/{id}/Schedules	POST	Creates a schedule for the built-in report with the specified report ID.	POST Reports ID Schedules on page 1173
/{id}/Schedules	PUT	Updates a schedule for the built-in report with the specified report ID.	PUT Reports ID Schedules on page 1183

2.6.28.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/read/

Table 442: 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 1157) to retrieve a list of your built-in reports to determine the report ID to use.

Table 443: 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 464). 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>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>at</td><td>report run. Supported values vary depending on the selected report and include:
PDF
Excel
CSV</td></tr>
<tr><td>KeyfactorSchedule</td><td>An object providing the schedule for the report. The schedule can be one of:
<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:</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name — Description **Nam-e** — **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 repres-enting 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" }` Mont-hly — 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>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>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>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:</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/modify/

Table 444: 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 1163) to retrieve a list of your custom report links to determine the report ID to use.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/read/

Table 445: 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 1163) to retrieve a list of your custom reports to determine the report ID to use.

Table 446: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/modify/

Table 447: 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 1157) 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 1137) to retrieve the details for that report to determine the schedule ID to use.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.5 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 1168).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/read/

Table 448: 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 1157) 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 1137) to retrieve the details for that report to determine the schedule ID to use.

Table 449: 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:

<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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/read/

Table 450: 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 1157) to retrieve a list of your built-in reports to determine the report ID to use.

Table 451: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.7 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/modify/

Table 452: 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 1157) 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 1137) 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 453: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.8 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 1137).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/read/

Table 454: 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* in the *Keyfactor Command Reference Guide*. 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 455: 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 464). 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.9 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 1173) or *PUT /Reports/{id}Schedules* (see PUT Reports ID Schedules on page 1183) method. To update parameters for a built-in report, use the *PUT /Reports/{id}/Parameters* method (see PUT Reports ID Parameters on page 1155). This method returns HTTP 200 OK on a success with the details of the report.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/modify/

Table 456: 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 1157) 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 464). 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 457: 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 464). 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.10 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/read/

Table 458: 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* in the *Keyfactor Command Reference Guide*. 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 459: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.11 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/modify/

Table 460: 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 461: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.12 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/modify/

Table 462: 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 1163) 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 463: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.13 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 1148).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/read/

Table 464: 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 1157) to retrieve a list of your built-in reports to determine the report ID to use.

Table 465: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.14 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/modify/

Table 466: 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 1157) 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>

<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. 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</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 467: 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:

<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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.28.15 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/reports/modify/

Table 468: 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 1157) 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 1137) 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>

```
"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>

<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 469: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.29 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 470: Scheduling Endpoints

Endpoint	Method	Description	Link
/	POST	Create or update task management schedules.	

2.6.29.1 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 134, PUT Alerts Expiration Schedule on page 95, PUT Alerts Key Rotation Schedule on page 169, and PUT Alerts Pending Schedule on page 204).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/modify/

Table 471: POST Scheduling 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 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 <a>GET Certificate Authority Health Monitoring Schedule on page 451) to retrieve the schedule for CA health monitoring to determine the health monitoring schedule ID.</td></tr>
<tr><td>ScheduleType</td><td>Body</td><td>Required. An integer indicating the type of schedule to be updated. Supported schedule types are:
<table><tr><th>Code</th><th>Category Name</th></tr><tr><td>1</td><td>Expiration Alert</td></tr><tr><td>2</td><td>Pending Alert</td></tr><tr><td>10</td><td>CA Health Monitoring Alert</td></tr><tr><td>20</td><td>Issued Alert</td></tr><tr><td>22</td><td>SSH Key Rotation Alert</td></tr></table></td></tr>
<tr><td>Enabled</td><td>Body</td><td>A Boolean that indicates whether the schedule is enabled (true) or not (false). The default is false.</td></tr>
<tr><td>Interval</td><td>Body</td><td>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.</td></tr>
<tr><td>TimeOfDay</td><td>Body</td><td>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.</td></tr>
</table>

Table 472: POST Scheduling 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 schedule.</td></tr>
<tr><td>Schedule</td><td>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).</td></tr>
<tr><td>ScheduleType</td><td>An integer indicating the type of schedule. Supported schedule types are:<table><tr><th>Code</th><th>Category Name</th></tr><tr><td>1</td><td>Expiration Alert</td></tr><tr><td>2</td><td>Pending Alert</td></tr><tr><td>10</td><td>CA Health Monitoring Alert</td></tr><tr><td>20</td><td>Issued Alert</td></tr><tr><td>22</td><td>SSH Key Rotation Alert</td></tr></table></td></tr>
<tr><td>Enabled</td><td>A Boolean that indicates whether the schedule is enabled (true) or not (false).</td></tr>
<tr><td>Name</td><td>A string indicating the type of job.</td></tr>
<tr><td>EntityId</td><td>This is considered deprecated and may be removed in a future release.</td></tr>
<tr><td>LastRun</td><td>This is considered deprecated and may be removed in a future release.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30 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 473: Security Endpoints

Endpoint	Method	Description	Link
/Identities	POST	Adds a new security identity into Keyfactor Command.	POST Security Identities on page 1207
/Identities	GET	Returns all security identities with with sorting and filter options.	GET Security Identities on page 1203
/Identities/Lookup	GET	Validates that the identity with the specified name exists.	GET Security Identities Lookup on page 1202
/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 1209
/Containers/{id}/Roles	POST	Sets the permissions of the certificate store container with the specified ID.	POST Security Containers ID Roles on page 1210
/Audit/Collections/{id}	GET	Returns permission details for the certificate collection with the specified ID.	GET Security Audit Collections ID on page 1211
/My	GET	Returns permission details for the current user, including certificate collections and certificate store containers.	GET Security My on page 1214

2.6.30.1 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 1203) 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 1196).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/modify/

Table 474: 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 1203) to retrieve a list of all the security identities to determine the identity's ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30.2 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 1196).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 475: 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 1203) to retrieve a list of all the security identities to determine the identity's ID.

Table 476: 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:

KEYEXAMPLE\\PKI Administrators</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 Security Roles and Claims in the Keyfactor Command Reference Guide.</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 Certificate Permissions in the Keyfactor Command Reference Guide.
</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 in the *Keyfactor Command Reference Guide*.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30.3 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 1207) 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 477: 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 478: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30.4 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 1196).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/modify/

Table 479: 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 480: 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 1258) 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 1128).</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 Security Role Operations: Version One Permission Model in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30.5 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 1202) 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 1196).

Tip: This method cannot be used to assign roles to an identity. Use the *PUT /Security/Roles* method (see PUT Security Roles on page 1275) to assign roles to an identity.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 481: 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 482: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30.6 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 1238 to list permissions on certificate store containers for a specified security role.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 483: 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 643) to determine the ID of the certificate store container you wish to evaluate.

Table 484: 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 Security Role Operations: Certificate Store Management in the *Keyfactor Command Reference Guide* for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30.7 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 1239 to assign permissions for one or more certificate store containers to a security role.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 485: 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 643) 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 1258) 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 Security Role Operations: Certificate Store Management in the *Keyfactor Command Reference Guide* for an overview of the possible permissions. For example: `"Permissions": [ "Read", "Modify" ]`

Table 486: 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 Security Role Operations: Certificate Store Management in the *Keyfactor Command Reference Guide* for an overview of the possible permissions.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30.8 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 1243 to list permissions on certificate collections for a specified security role.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 487: 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 461) to determine the ID of the certificate collection you wish to evaluate.

Table 488: 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 Security Role Operations: Version One Permission Model in the *Keyfactor Command Reference Guide* 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,
```

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

<pre>
        41
      ]
   },
   {
      "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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.30.9 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
None

Table 489: 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 Security Role Operations: Version One Permission Model in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.31 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 490: 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 1225
/	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 1218
/	PUT	Updates the security claim with the specified ID.	PUT Security Claims on page 1222
/{id}	DELETE	Deletes the security claim with the specified ID.	DELETE Security Claims ID on page 1227
/Roles	GET	Returns the security claim assigned to the security claim identified by the selected parameters.	GET Security Claims Roles on page 1227

2.6.31.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 491: 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* in the *Keyfactor Command Reference Guide*. 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 492: 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

Name	Description
	Command reference GUID for the provider.
AuthenticationScheme	A string indicating the provider auth scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).
DisplayName	A string containing the short reference name for the provider (e.g. Active Directory).

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.31.2 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 493: 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 494: 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>

Name	Description
	<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>
ProviderAuthenticationScheme	A string indicating the provider authentication scheme (e.g. Active Directory, or Client Certificate Authentication CA, or unknown).

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.31.3 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 495: 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 496: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.31.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 497: 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 1215) to retrieve a list of all the security claims to determine the claim's ID.

Table 498: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.31.5 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 499: 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 1215) to determine the ID of the security claim you wish to delete.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.31.6 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 1225 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 500: 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 1215 or GET Security Claims ID on page 1225.

Table 501: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32 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 502: 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 1233
/{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 1234
/{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 1236
/{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 1238
/{id}/Permisssions/Containers	POST	Adds container permissions to the security role that matches the ID.	POST Security Roles ID Permissions Containers on page 1239
/{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 1241
/{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 1243
/{id}/Permisssions/Collections	POST	Adds collection permissions to the security role that matches the ID.	POST Security Roles ID Permissions Collections on page 1244
/{id}/Permisssions/Collections	PUT	Sets collection permissions to the security role that matches the ID.	PUT Security Roles ID Permissions Collections on page 1246
/{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 1248
/{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 1249

2.6.32.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 503: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 504: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 505: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 506: 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: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.3 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 507: 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 1258) 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	Description
Area	**Required**. A string indicating the name of the permissions to grant (e.g. AdminPortal).
Permission	**Required**. A string indicating the permission level to grant (e.g. Read)

Table 508: 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	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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.4 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 509: 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 1258) 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 510: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 511: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 512: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.6 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 513: 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 1258) 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 514: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.7 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 515: 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 1258) 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 516: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.8 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 517: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 518: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.9 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 519: 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 1258) 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 520: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.10 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 521: 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 1258) 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 522: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.11 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/read/

Table 523: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 524: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.32.12 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/security/modify/

Table 525: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 526: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.33 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 527: 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 below
/	GET	Returns all security roles with filtering and output options.	GET Security Roles on page 1258
/	POST	Adds a new security role.	POST Security Roles on page 1263
/	PUT	Updates the security role with the specified ID.	PUT Security Roles on page 1275
/{id}/Identities	GET	Returns the security identities assigned to the security role with the specified ID.	GET Security Roles ID Identities on page 1292
/{id}/Identities	PUT	Updates the security identities assigned to the security role with the specified ID.	PUT Security Roles ID Identities on page 1290
/{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 1287

2.6.33.1 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 1258) 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* in the *Keyfactor Command Reference Guide*) 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 1128).

Table 528: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.33.2 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* in the *Keyfactor Command Reference Guide*) 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 11.

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 529: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 530: 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 1128).
Permissions	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Security Role Operations: Version Two Permission Model in the *Keyfactor Command Reference Guide* 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 531: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 532: 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 1128).
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 Security Role Operations: Version One Permission Model in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.33.3 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* in the *Keyfactor Command Reference Guide*) 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 11.

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 533: 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* in the *Keyfactor Command Reference Guide*. 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 534: 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 1128). **Tip:** For details of the permissions associated with the role, use the *GET /SecurityRoles/{id}* method (see GET Security Roles ID on page 1252) 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 1130) 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 1252) 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 535: 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* in the *Keyfactor Command Reference Guide*. 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 536: 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 1128).
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 Security Role Operations: Version One Permission Model in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.33.4 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* in the *Keyfactor Command Reference Guide*) 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 1128).

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 11.

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 537: 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 1128).
Permissions	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Security Role Operations: Version Two Permission Model in the *Keyfactor Command Reference Guide* 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 538: 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 1128).
Permissions	Body	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Security Role Operations: Version Two Permission Model in the *Keyfactor Command Reference Guide* 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:

<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>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>
</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 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 539: 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 1128).</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 Security Role Operations: Version One Permission Model in the Keyfactor Command Reference Guide 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</td></tr></table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>* 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 540: 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 1128).
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 Security Role Operations: Version One Permission Model in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.33.5 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* in the *Keyfactor Command Reference Guide*) 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 1128).

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 11.

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 541: 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 1258) 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 1128).
Permissions	Body	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Security Role Operations: Version Two Permission Model in the *Keyfactor Command Reference Guide* 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)

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 542: 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 1128).
Permissions	Body	An array of strings containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. See Security Role Operations: Version Two Permission Model in the *Keyfactor Command Reference Guide* 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:

<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>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>
</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>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>
</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 543: PUT Security Roles 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 containing the Keyfactor Command identifier for the security role.
Use the GET /Security/Roles method (see <a>GET Security Roles on page 1258) to retrieve a list of all the security roles to determine the role's ID.</td></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 1128).</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 Security Role Operations: Version One Permission Model in the Keyfactor Command Reference Guide 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</td></tr></table></td></tr>
</table>

Name	In	Description

		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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 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 544: 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 1128).
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 Security Role Operations: Version One Permission Model in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.33.6 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 *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 1128).

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 1252 and POST Security Roles on page 1263).

Table 545: 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 1258) 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 546: 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 1128).
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 Security Role Operations: Version One Permission Model in the Keyfactor Command Reference Guide for an overview of the possible permissions.
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre></td></tr>
</table>

Tip: See the interactive code examples installed with your Keyfactor Command instance in the Keyfactor API Reference and Utility. *Keyfactor API Reference and Utility*.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.33.7 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* in the *Keyfactor Command Reference Guide*) 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 1128).

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 1275).

Table 547: 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 1258) 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 1203) to retrieve a list of all the security identities to determine the identity ID(s).

Table 548: 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 interactive code examples installed with your Keyfactor Command instance in the Keyfactor API Reference and Utility. *Keyfactor API Reference and Utility*.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.33.8 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* in the *Keyfactor Command Reference Guide*) 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 1252).

Table 549: 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 1258) to retrieve a list of all the security roles to determine the role's ID.

Table 550: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.34 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 551: SSH Endpoints

Endpoint	Method	Description	Link
/Keys/Unmanaged/{id}	DELETE	Delete a discovered unmanaged SSH key for the specified ID.	DELETE SSH Keys Unmanaged ID on page 1298
/Keys/Unmanaged/{id}	GET	Retrieve details for a discovered unmanaged SSH key for the specified ID.	GET SSH Keys Unmanaged ID on page 1299
/Keys/MyKey	GET	Retrieve details for a user's SSH key generated through Keyfactor Command.	GET SSH Keys My Key on page 1300
/Keys/MyKey	POST	Generate a new SSH key pair for a user through Keyfactor Command.	POST SSH Keys My Key on page 1303
/Keys/MyKey	PUT	Update an SSH key for a user through Keyfactor Command.	PUT SSH Keys My Key on page 1307
/Keys/Unmanaged	DELETE	Delete one or more discovered unmanaged SSH keys based on a selection query.	DELETE SSH Keys Unmanaged on page 1310
/Keys/Unmanaged	GET	Retrieve details for one or more discovered unmanaged SSH keys based on a selection query.	GET SSH Keys Unmanaged on page 1311
/Logons/{id}	DELETE	Deletes a Linux logon from Keyfactor Command.	DELETE SSH Logons ID on page 1314
/Logons/{id}	GET	Returns information about a Linux logons.	GET SSH Logons ID on page 1315
/Logons/	GET	Returns information about one or	GET SSH

Endpoint	Method	Description	Link
		more Linux logons.	Logons on page 1317
/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 1319
/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 1322
/Servers/{id}	DELETE	Deletes the SSH server with the specified ID.	DELETE SSH Servers ID on page 1324
/Servers/{id}	GET	Returns the SSH server with the specified ID.	GET SSH Servers ID on page 1325
/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 1329
/Servers/	GET	Returns a list of a SSH servers configured in Keyfactor Command.	GET SSH Servers on page 1332
/Servers/	POST	Creates a new SSH server.	POST SSH Servers on page 1337
/Servers/	PUT	Updates an existing SSH server.	PUT SSH Servers on page 1342
/Servers/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server.	DELETE SSH Servers Access on page 1347
/Servers/Access	POST	Creates Linux logon to user and service account mappings for an SSH	POST SSH Servers

Endpoint	Method	Description	Link
		server.	Access on page 1351
/ServerGroups/{id}	DELETE	Deletes the SSH server group with the specified ID.	DELETE SSH Server Groups ID on page 1355
/ServerGroups/{id}	GET	Returns the SSH server group with the specified ID.	GET SSH Server Groups ID on page 1355
/ServerGroups/{name}	GET	Returns the SSH server group with the specified name.	GET SSH Server Groups Name on page 1360
/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 1364
/ServerGroups/	GET	Returns a list of a SSH server groups configured in Keyfactor Command.	GET SSH Server Groups on page 1367
/ServerGroups/	POST	Creates a new SSH server group.	POST SSH Server Groups on page 1372
/ServerGroups/	PUT	Updates an existing SSH server group.	PUT SSH Server Groups on page 1380
/ServerGroups/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server group.	DELETE SSH Server Groups Access on page 1388
/ServerGroups/Access	POST	Creates Linux logon to user and service account mappings for an SSH server group.	POST SSH Server Groups Access on page 1391

Endpoint	Method	Description	Link
/ServiceAccounts/{id}	DELETE	Deletes the SSH service account with the specified ID.	DELETE SSH Service Accounts ID on page 1394
/ServiceAccounts/{id}	GET	Returns the SSH service account with the specified ID.	GET SSH Service Accounts ID on page 1397
/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 1404
/ServiceAccounts/	DELETE	Deletes one or more SSH service accounts with the specified IDs.	DELETE SSH Service Accounts on page 1408
/ServiceAccounts/	GET	Returns a list of SSH service accounts based on the specified filters.	GET SSH Service Accounts on page 1410
/ServiceAccounts/	POST	Creates a new SSH service account.	POST SSH Service Accounts on page 1418
/ServiceAccounts/	PUT	Updates an existing SSH service account.	PUT SSH Service Accounts on page 1428
/ServiceAccounts/Rotate/{id}	POST	Generates a new key pair for an existing service account.	POST SSH Service Accounts Rotate ID on page 1436
/Users/{id}	DELETE	Deletes the SSH user with the specified ID.	DELETE SSH Users ID on page 1440

Endpoint	Method	Description	Link
/Users/{id}	GET	Returns the SSH user with the specified ID.	GET SSH Users ID on page 1441
/Users/	GET	Returns a list of SSH users based on the specified filters.	GET SSH Users on page 1446
/Users/	POST	Creates a new SSH user.	POST SSH Users on page 1457
/Users/	PUT	Updates an existing SSH user.	PUT SSH Users on page 1459
/Users/Access	POST	Creates a mapping from the SSH user to one or more Linux logons.	POST SSH Users Access on page 1461

2.6.34.1 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* 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 552: 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 1299
/MyKey	GET	Retrieve details for a user's SSH key generated through Keyfactor Command.	GET SSH Keys My Key on page 1300
/MyKey	POST	Generate a new SSH key pair for a user through Keyfactor Command.	POST SSH Keys My Key on page 1303

Endpoint	Method	Description	Link
/MyKey	PUT	Update an SSH key for a user through Keyfactor Command.	PUT SSH Keys My Key on page 1307
Unmanaged	DELETE	Delete one or more discovered unmanaged SSH keys based on a selection query.	DELETE SSH Keys Unmanaged on page 1310
Unmanaged	GET	Retrieve details for one or more discovered unmanaged SSH keys based on a selection query.	GET SSH Keys Unmanaged on page 1311

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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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* in the *Keyfactor Command Reference Guide* for more information.

Table 553: 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 1311) to retrieve a list of all the unmanaged keys to determine the unmanaged key's ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 554: 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 1311) to retrieve a list of all the unmanaged keys to determine the unmanaged key's ID.

Table 555: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 1303). This method returns HTTP 200 OK on a success with the key's details.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssh/user/
OR
/ssh/server_admin/
OR
/ssh/enterprise_admin/

Table 556: 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 557: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 1322, POST SSH Server Groups Access on page 1391, and POST SSH Servers Access on page 1351). This method returns HTTP 200 OK on a success with the key's details.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssh/user/
OR
/ssh/server_admin/
OR
/ssh/enterprise_admin/

Table 558: 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 1300) 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 559: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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 560: 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 561: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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* in the *Keyfactor Command Reference Guide* for more information.

Table 562: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 563: 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* in the *Keyfactor Command Reference Guide*. 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 564: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.34.2 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 565: 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 1317
/	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 1319
/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 1322

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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 566: 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 1317) to retrieve a list of all the SSH logons to determine the logon's ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 567: 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 1317) to retrieve a list of all the SSH logons to determine the logon's ID.

Table 568: 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 SSH Servers in the Keyfactor Command Reference Guide 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 SSH Server Groups in the Keyfactor Command Reference Guide 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 SSH in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 569: 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* in the *Keyfactor Command Reference Guide*. 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 570: 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* in the *Keyfactor Command Reference Guide* for more information.
GroupName	A string indicating the server group to which the server referenced by *ServerName* belongs. See *SSH Server Groups* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 571: 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 1332) 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* 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 1446) to retrieve a list of all the users (including service accounts) created in Keyfactor Command to determine a user's ID.

Table 572: 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 SSH Servers in the Keyfactor Command Reference Guide 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 SSH Server Groups in the Keyfactor Command Reference Guide 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 SSH in the Keyfactor Command Reference Guide 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 573: 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 1317) 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 1446) to retrieve a list of all the users (including service accounts) created in Keyfactor Command to determine a user's ID. See *SSH* in the *Keyfactor Command Reference Guide* for more information about users and service accounts.

Table 574: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.34.3 SSH Servers

The SSH Servers component of the Keyfactor API includes methods necessary to create, update, and delete SSH servers within Keyfactor Command.

Table 575: 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 1325

Endpoint	Method	Description	Link
/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 1329
/	GET	Returns a list of a SSH servers configured in Keyfactor Command.	GET SSH Servers on page 1332
/	POST	Creates a new SSH server.	POST SSH Servers on page 1337
/	PUT	Updates an existing SSH server.	PUT SSH Servers on page 1342
/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server.	DELETE SSH Servers Access on page 1347
/Access	POST	Creates Linux logon to user and service account mappings for an SSH server.	POST SSH Servers Access on page 1351

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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 576: 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 1332) to retrieve a list of all the SSH servers to determine the server's ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 577: 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 1332) to retrieve a list of all the SSH servers to determine the server's ID.

Table 578: 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 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 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 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 579: 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 1332) to retrieve a list of all the SSH servers to determine the server's ID.

Table 580: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 581: 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* in the *Keyfactor Command Reference Guide*. 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 582: 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 SSH Server Groups 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 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 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 1372) and that your Keyfactor Bash Orchestrator has been registered and approved in Keyfactor Command (see GET Agents on page 17).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 583: 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 584: 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 SSH Server Groups 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 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 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 585: 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 586: 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 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 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 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 587: 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 588: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 589: 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 590: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.34.4 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 591: 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 1360
/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 1364
/	GET	Returns a list of a SSH server groups configured in Keyfactor Command.	GET SSH Server Groups on page 1367
/	POST	Creates a new SSH server group.	POST SSH Server Groups on page 1372
/	PUT	Updates an existing SSH server group.	PUT SSH Server Groups on page 1380
/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server group.	DELETE SSH Server Groups Access on page 1388
/Access	POST	Creates Linux logon to user and service account mappings for an SSH server group.	POST SSH Server Groups Access on page 1391

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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssh/enterprise_admin/

Table 592: 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 1367) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 593: 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 1367) to retrieve a list of all the SSH server groups to determine the server group's GUID.

Table 594: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 595: 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 596: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 597: 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 1367) to retrieve a list of all the SSH server groups to determine the server group's ID.

Table 598: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 599: 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* in the *Keyfactor Command Reference Guide*. 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 600: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssh/enterprise_admin/

Table 601: 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* 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 602: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 603: 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 SSH Server Groups 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 604: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 605: DELETE 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. An integer indicating the Keyfactor Command reference ID for the SSH server group.</td></tr>
<tr><td>LogonUsers</td><td>Body</td><td>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 removed from association with the logon.</td></tr></table>For example:
<pre>"LogonUsers": [
   {
      "LogonName": "johns",
      "Users": [
         "KEYEXAMPLE\\jsmith"
      ]
   }
]</pre></td></tr>
</table>

Table 606: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 607: 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 608: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.34.5 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 609: 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 1397
/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 1404
/	DELETE	Deletes one or more SSH service accounts with the specified IDs.	DELETE SSH Service Accounts on page 1408
/	GET	Returns a list of SSH service accounts based on the specified filters.	GET SSH Service Accounts on page 1410
/	POST	Creates a new SSH service account.	POST SSH Service Accounts on page 1418
/	PUT	Updates an existing SSH service account.	PUT SSH Service Accounts on page 1428
/Rotate/{id}	POST	Generates a new key pair for an existing service account.	POST SSH Service Accounts Rotate ID on page 1436

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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 610: 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 1410) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 1404).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 611: 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 1410) to retrieve a list of all the SSH service accounts to determine the service account's ID.

Table 612: 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* 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* 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

	Name: Description
	Nam-e: Description
	Off: Turn off a previously configured schedule.
	Interv-al: 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:m-m: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>
<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

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH service account user.
Key	An object containing information about the key for the service account user. Key details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH service account'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 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

<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 Application Settings: SSH Tab 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. myap-p@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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 613: 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 1410) 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 614: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 615: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 1404).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 616: 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* in the *Keyfactor Command Reference Guide*. 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 617: 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.
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* 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* 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

	Name: Description
	Nam-e: Description
	Off: Turn off a previously configured schedule.
	Interv-al: 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:m-m: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>
<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 Application Settings: SSH Tab 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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 1372) and that your Keyfactor Bash Orchestrator has been registered and approved in Keyfactor Command (see GET Agents on page 17).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 618: 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 in the Keyfactor Command Reference Guide for more information.</td></tr>
</table>

Table 619: 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.
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* 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* 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

	Name: Description
	Nam-e: Description
	Off: Turn off a previously configured schedule.
	Interv-al: 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:m-m: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>
<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 Application Settings: SSH Tab 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. myap-p@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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 620: 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 1410) to retrieve a list of all the SSH service accounts to determine the service account's ID.</td></tr>
</table>

Table 621: 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.
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* 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* 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

	Name: Description
	Nam-e: Description
	Off: Turn off a previously configured schedule.
	Interv-al: 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:m-m:ss.000Z (e.g. 2023-11-19T16:23:01Z).
	For example, daily at 11:30 pm:
	`"Daily": {`

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

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH service account user.
Key	An object containing information about the key for the service account user. Key details include:

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the SSH service account'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 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

<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 Application Settings: SSH Tab 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. myap-p@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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 622: 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 1410) 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 623: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.34.6 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 624: 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 1446
/	POST	Creates a new SSH user.	POST SSH Users on page 1457
/	PUT	Updates an existing SSH user.	PUT SSH Users on page 1459
/Access	POST	Creates a mapping from the SSH user to one or more Linux logons.	POST SSH Users Access on page 1461

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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssh/server_admin/

Table 625: 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 1446) to retrieve a list of all the SSH users to determine the user's ID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence 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* 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 1300) for a user account or the GET /SSH/ServiceAccounts/Key/{id} method (see GET SSH Service Accounts Key ID on page 1404) for a service account.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 11.

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 626: 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 1446) to retrieve a list of all the SSH users to determine the user's ID.

Table 627: 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* 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 628: 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 1446) to retrieve a list of all the SSH users to determine the user's ID.

Table 629: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* 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 1300) for user accounts and the GET /SSH/ServiceAccounts/Key/{id} method (see GET SSH Service Accounts Key ID on page 1404) for service accounts.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 11.

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 630: 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 1332) or the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 1367) 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* in the *Keyfactor Command Reference Guide*. 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 631: 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* 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 632: 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 1332) or the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 1367) 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* in the *Keyfactor Command Reference Guide*. 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 633: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 634: 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 1317) to retrieve a list of all the SSH logons to determine the logon's ID(s).

Table 635: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

Table 636: 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 1446) 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 1317) 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 1446) before updating and provide both existing and new logon IDs.

Table 637: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference 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* in the *Keyfactor Command Reference Guide*) 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* in the *Keyfactor Command Reference Guide*.

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 638: 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 1446) 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 1317) 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 1446) before updating and provide both existing and new logon IDs.

Table 639: 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* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Refer-

ence and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.35 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 640: 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 1466
/Test	POST	Sends a test email message to confirm SMTP configuration.	POST SMTP Test on page 1469

2.6.35.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/system_settings/read/

Table 641: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.35.2 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* in the *Keyfactor Command Reference Guide*) 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 642: 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 643: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.35.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/system_setting/modify/

Table 644: 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 645: 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*.
TestRecipient	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	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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36 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 646: 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 1475
/Endpoints/{id}	GET	Returns the details about a single endpoint discovered during SSL scanning.	GET SSL Endpoints ID on page 1477
/NetworkRanges/{id}	DELETE	Removes all network ranges from the specified SSL network.	DELETE SSL NetworkRanges ID on page 1479
/NetworkRanges/{id}	GET	Returns network range information about the specified SSL network.	GET SSL NetworkRanges ID on page 1479
/Networks/{identifier}	GET	Returns information about the specified SSL network.	GET SSL Networks Identifier on page 1480
/	GET	Returns the results of an SSL scan based on query information.	GET SSL on page 1491
/Networks	GET	Returns information about all SSL networks in Keyfactor Command.	GET SSL Networks on page 1493
/Networks	POST	Creates a new SSL network.	POST SSL Networks on page 1504
/Networks	PUT	Updates an existing SSL	PUT SSL Networks

Endpoint	Method	Description	Link
		network.	on page 1518
/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 1532
/Networks/{id}/Parts	GET	Returns the scan job information for SSL discovery or monitoring.	GET SSL Networks ID Parts on page 1537
/NetworkRanges	POST	Adds network ranges to the specified SSL network.	POST SSL NetworkRanges on page 1539
/NetworkRanges	PUT	Updates network range information on the specified SSL network.	PUT SSL NetworkRanges on page 1540
/Endpoints/ReviewStatus	PUT	Used to change the *reviewed* status for a given SSL endpoint.	PUT SSL Endpoints Review Status on page 1541
/Endpoints/MonitorStatus	PUT	Used to change the *monitoring* status for a given SSL endpoint.	PUT SSL Endpoints Monitor Status on page 1542
/Endpoints/ReviewAll	PUT	Used to change the *reviewed* status for all given SSL endpoints to true.	PUT SSL Endpoints Review All on page 1542
/Endpoints/MonitorAll	PUT	Used to change the *monitoring* status for all given SSL endpoints to true.	PUT SSL Endpoints Monitor All on page 1543
/Networks/{id}/Scan	POST	Starts an SSL discovery or monitoring scan job manually.	POST SSL Networks ID Scan on page 1544
/NetworkRanges/Validate	POST	Validates all SSL networks given.	POST SSL NetworkRanges Validate on page 1545
/Networks/{id}	DELETE	Removes an SSL network from Keyfactor Command.	DELETE SSL Networks ID on page 1546

2.6.36.1 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 1537). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 647: 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 1537) to retrieve a list of all the scan job segments in an SSL network to determine the SSL scan job segment's GUID.

Table 648: GET SSL Parts {id} Response Data

Parameter Name	Description
ScanJobPartId	A string indicating the Keyfactor Command reference GUID for the scan job segment.
LogicalScanJobId	A string indicating the Keyfactor Command reference GUID for the scan job as a whole.
AgentJobId	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.
EstimatedEndpointCount	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* in the *Keyfactor Command Reference Guide*).
Status	An integer indicating the status of the scan job segment. Possible values are: **Value** / **Description** 1 / Not Started 2 / In Progress 3 / Complete
StatTotalEndpointCount	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.
StatTimedOutConnectingCount	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.
StatConnectionRefusedCount	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.
StatTimedOutDownloadingCount	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.

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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.2 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 649: 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 1491) to retrieve a list of all the SSL endpoints to determine the SSL endpoint's GUID.

Table 650: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.3 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* in the *Keyfactor Command Reference Guide*) 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 1518).

Table 651: 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 1493) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon () at the top of the Management Portal page next to the **Log Out** button.

2.6.36.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 652: 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 1493) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Table 653: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 654: 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 1493) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Table 655: 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* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 656: 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* in the *Keyfactor Command Reference Guide*. 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 657: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.7 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 658: 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* in the *Keyfactor Command Reference Guide*. 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 659: 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* in the *Keyfactor Command Reference Guide* 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.
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API

Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.8 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/modify/

Table 660: 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>

```
}
```

</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).
DiscovererTimeoutMs	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

		"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"
		}
		]

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 661: 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* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.9 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* in the *Keyfactor Command Reference Guide*) 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 662: 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 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>DiscoverPercentComplete</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).
DiscovererTimeoutMs	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 663: 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* in the *Keyfactor Command Reference Guide* 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.10 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 664: 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 1491) 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 665: GET SSL Endpoints {id} History Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>HistoryId</td><td>A string indicating the Keyfactor Command reference GUID for the history entry.</td></tr>
<tr><td>EndpointId</td><td>A string indicating the Keyfactor Command reference GUID for the endpoint with which the history is associated.</td></tr>
<tr><td>AuditId</td><td>An integer indicating the Keyfactor Command ID used to track progress during scan jobs.</td></tr>
<tr><td>Timestamp</td><td>A string indicating the date and time the history entry was created.</td></tr>
<tr><td>Status</td><td>An integer containing the status of the scan for which the history item was created. 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>TimeOutConnecting</td></tr>
<tr><td>2</td><td>ExceptionConnecting</td></tr>
<tr><td>3</td><td>TimeoutDownloading</td></tr>
<tr><td>4</td><td>ExceptionDownloading</td></tr>
<tr><td>5</td><td>NotSsl</td></tr>
<tr><td>6</td><td>CertificateFound</td></tr>
<tr><td>7</td><td>ExceptionInSql</td></tr>
<tr><td>8</td><td>InvalidOrUnreachableHost</td></tr>
<tr><td>9</td><td>ConnectionRefused</td></tr>
<tr><td>10</td><td>BadSslHandshake</td></tr>
<tr><td>11</td><td>ClientAuthenticationFailed</td></tr>
<tr><td>12</td><td>NoCertificate</td></tr>
<tr><td>13</td><td>SslRefused</td></tr>
<tr><td>14</td><td>NotProbed</td></tr>
</table></td></tr>
<tr><td>JobType</td><td>An integer containing the type of scan job from which the history entry was created.</td></tr>
</table>

<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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.11 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* in the *Keyfactor Command Reference Guide*). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 666: 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 1493) 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* in the *Keyfactor Command Reference Guide*. 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 667: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.12 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/modify/

Table 668: 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 1493) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.13 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* in the *Keyfactor Command Reference Guide*) 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 669: 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 1493) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.14 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/modify/

Table 670: 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 1491) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.15 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/modify/

Table 671: 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 1491) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.16 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:

/ssl/modify/

Table 672: 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* in the *Keyfactor Command Reference Guide*.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.17 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/modify/

Table 673: 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* in the *Keyfactor Command Reference Guide*.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.18 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/modify/

Table 674: 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 1493) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.19 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/modify/

Table 675: 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 1493) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.20 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/read/

Table 676: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.36.21 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/ssl/modify/

Table 677: 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 1493) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.37 Status

The Status component of the Keyfactor API includes methods necessary to retrieve the current list of Keyfactor API endpoints.

Table 678: Status Endpoints

Endpoint	Method	Description	Link
/Endpoints	GET	Returns a list of the Keyfactor API endpoints.	GET Status Endpoints below

2.6.37.1 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 8). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
None

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.38 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 679: 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

Endpoint	Method	Description	Link
			Settings on page 1566
/Settings	PUT	Sets global values for template policy.	PUT Templates Settings on page 1573
/SubjectParts	GET	Returns a list of supported subject parts for template regular expressions and default subjects.	GET Templates Subject Parts on page 1592
/	GET	Returns a list of templates.	GET Templates on page 1593
/	PUT	Updates selected settings for the specified template.	PUT Templates on page 1606
/Import	POST	Import templates from a specified configuration tenant into Keyfactor Command	POST Templates Import on page 1642

2.6.38.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_templates/read/

Table 680: 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 1593) to retrieve a list of all the templates to determine the template ID.

Table 681: 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>

<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 in the Keyfactor Command Reference Guide 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 1566. 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 1592) 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 1566. 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 1592) 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 1566. 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>

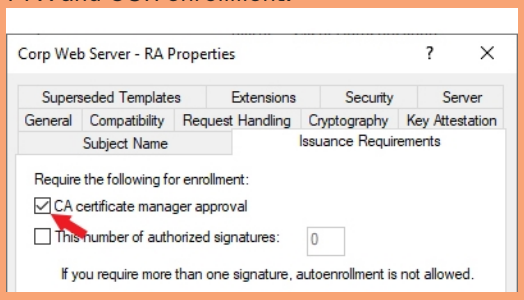
Name	Description

Value	Description
	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.
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

<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* in the *Keyfactor Command Reference Guide* 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 8: 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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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</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>Active Directory.</td></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 and Certificate Authority Operations in the Keyfactor Command Reference Guide). For more information about one-click renewals, see Certificate Operations: Renew in the Keyfactor Command Reference Guide.</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.38.2 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 8).

Tip: Template policies may also be set at an individual template level to apply to a single template (see PUT Templates on page 1606). 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_templates/read/

Table 682: 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 1592) 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**:

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 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.38.3 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 1606). 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* in the *Keyfactor Command Reference Guide*) 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 683: 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 1592) 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**

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**
	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* 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 template along with the bit lengths and/or curves for the

Name	Description
	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. 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”).
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. Keyfactor Command supports key sizes 2048, 3072, 4096, 6144, 8192, and 16384.

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 684: 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 1592) 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**
	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>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>

<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 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:
<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</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>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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.38.4 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 8).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_templates/read/

Table 685: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.38.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_templates/read/

Table 686: 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* in the *Keyfactor Command Reference Guide*. 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 687: 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 in the Keyfactor Command Reference Guide 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 1566. 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 1592) 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**

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**

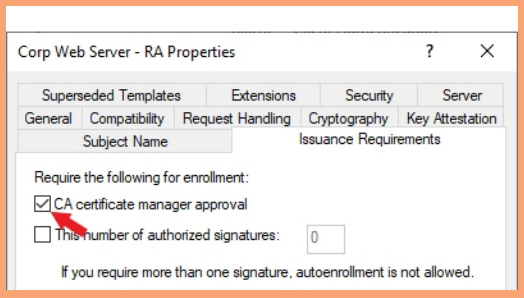
Subject Part	Example
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, 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:

<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-F0-9]{1,4}:){7}[A-F0-9]{1,4}$</pre></td></tr>
<tr><td>MAIL (Subject Alternative Name: Email)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the “@” made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly “@keyexample.com”:<pre>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</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>
</table>

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* in the *Keyfactor Command Reference Guide* 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 9: 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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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</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>Active Directory.</td></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 and Certificate Authority Operations in the Keyfactor Command Reference Guide). For more information about one-click renewals, see Certificate Operations: Renew in the Keyfactor Command Reference Guide.</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.38.6 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* in the *Keyfactor Command Reference Guide*) 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 688: 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 Adding or Modifying a CA Record in the Keyfactor Command Reference Guide for more information. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>None</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>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>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 GET Templates Settings on page 1566. 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 1592) 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**

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 1566. 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 1592) 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 GET Templates Settings on page 1566. 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* in the *Keyfactor Command Reference Guide* 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).

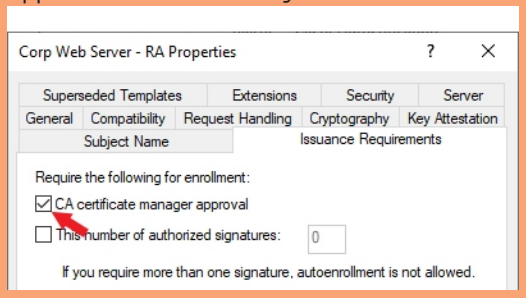
⚠ Important: Any templates that are configured on the Microsoft CA Issuance Requirements tab for CA certificate manager approval cannot be used for enrollment and associated alerting in Keyfactor Command without configuring private key retention. Any of the enabled private key retention settings (settings other than none as described for KeyRetention) will allow a template requiring manager approval to work with Keyfactor Command PFX and CSR enrollment.
 

Figure 10: 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 and Certificate Authority Operations in the Keyfactor Command Reference Guide). For more information about one-click renewals, see Certificate Oper-ations: Renew in the Keyfactor Command Reference Guide.</td></tr>
</table>

Table 689: 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 in the Keyfactor Command Reference Guide 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 1566. 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 1592) 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.

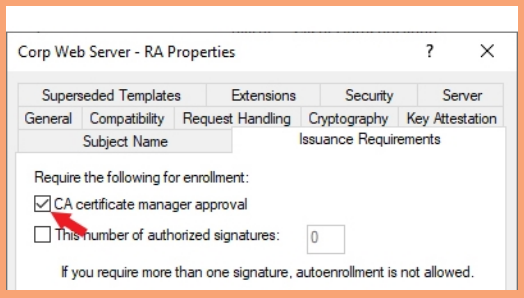
Note: The error message already includes a leading string with the subject part (e.g. “Common Name:” or “Invalid CN provided:” depending on the interface used). Your custom message follows this.</td></tr>
</table>
</td></tr>
<tr><td>TemplateDefaults</td><td>An array of objects containing individual template-level 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 1566. 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 1592) 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 1566. 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* in the *Keyfactor Command Reference Guide* 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 11: 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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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</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>Active Directory.</td></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 and Certificate Authority Operations in the Keyfactor Command Reference Guide). For more information about one-click renewals, see Certificate Operations: Renew in the Keyfactor Command Reference Guide.</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.38.7 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificate_templates/modify/

Table 690: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.39 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* in the *Keyfactor Command Reference Guide*) are not managed with these endpoints. Instead, refer to the Workflow Definitions and Workflow Instances endpoints (see Workflow Definitions on page 1659 and Workflow Instances on page 1805).

Table 691: 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

Endpoint	Method	Description	Link
/Certificates/Denied	GET	Retrieve a list of denied certificate request(s).	GET Workflow Certificates Denied on page 1647
/Certificates/Pending	GET	Retrieve a list of outstanding pending certificate request(s).	GET Workflow Certificates Pending on page 1650
/Certificates/ExternalValidation	GET	Retrieve a list of certificate request(s) requiring external validation.	GET Workflow Certificates External Validation on page 1653
/Certificates/Approve	POST	Approve a list of pending certificate request(s).	POST Workflow Certificates Approve on page 1658
/Certificates/Deny	POST	Deny a list of pending certificate request(s).	POST Workflow Certificates Deny on page 1656

2.6.39.1 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* 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 1659 and Workflow Instances on page 1805).

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 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 692: 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 1650) to retrieve a list of all the certificate requests to determine the certificate request ID.

Table 693: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.39.2 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* 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 1659 and Workflow Instances on page 1805).

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 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 694: 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* in the *Keyfactor Command Reference Guide*. 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 695: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.39.3 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* 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 1659 and Workflow Instances on page 1805).

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 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 696: 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* in the *Keyfactor Command Reference Guide*. 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 697: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.39.4 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* 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 1659 and Workflow Instances on page 1805).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/monitoring/alerts/read/

Table 698: 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* in the *Keyfactor Command Reference Guide*. 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 699: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.39.5 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* 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 1659 and Workflow Instances on page 1805).

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 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/requests/manage/

Table 700: 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 1650) 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 701: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.39.6 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* 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 1805).

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 in the *Keyfactor Command Reference Guide*).

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/certificates/requests/manage/

Table 702: 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 1650) to retrieve a list of all the certificate requests to determine the certificate request's IDs.

Table 703: POST Workflow Certificates Approve Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Successes</td><td>An array of strings indicating the successful approval 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 reason or description about why the request denials succeeded, failed or were denied.</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 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.</td></tr>
</table>

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40 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* in the *Keyfactor Command Reference Guide*.

Table 704: 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 1666
/{definitionId}	GET	Returns details of the workflow definition, including steps, for the workflow with the specified GUID.	GET Workflow Definitions Definition ID on page 1666
/{definitionId}	PUT	Updates the name and description of the workflow definition with the specified GUID.	PUT Workflow Definitions Definition ID on page 1691
/	GET	Returns a list of workflow definitions, without steps.	GET Workflow Definitions on page 1716
/	POST	Creates a new workflow definition, without steps.	POST Workflow Definitions on page 1719

Endpoint	Method	Description	Link
/Steps	GET	Returns information about the structure of the workflow definitions.	GET Workflow Definitions Steps on page 1745
/Types	GET	Returns a list of the defined workflow definition types.	GET Workflow Definitions Types on page 1751
/{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 1754
/{definitionId}/Publish	POST	Publishes the workflow definition with the specified GUID to activate it for use.	POST Workflow Definitions Definition ID Publish on page 1780

2.6.40.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/definitions/read/

Table 705: 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 1745) to retrieve a list of all the workflow definition steps to determine the extensionName.

Table 706: 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 *Extension/Scripts* in the *Keyfactor API*

Name	Description
	Reference Guide 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* in the *Keyfactor Command Reference Guide*. **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* in the *Keyfactor Command Reference Guide*). • **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

Name	Description
	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. ⚠ **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,

Name	Description
	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 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development.

It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.2 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* in the *Keyfactor Command Reference Guide*) 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 707: 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 1716) to retrieve a list of all the workflow definitions to determine the GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/definitions/read/

Table 708: 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 1716) 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 1250) 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 709: 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.

<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 *Extension/Scripts* in the *Keyfactor API Reference Guide* 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* in the *Keyfactor Command Reference Guide*. **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 Issued Request Alert Operations in the Keyfactor Command Reference Guide). - 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 </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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 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**</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>Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately 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 <a>SMTP on page 1464) 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 <a>Extensions Scripts on page 870).
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</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Value	Description
	`Platform \ExtensionLibrary\net6.0\Workflow`

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:AppOwnerFirstName)</td></tr>\n<tr><td>DN: $(re-

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	quest: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 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: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 envir-

Name	Description

Name	Description

Value	Description

onments 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 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

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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. • Load the secret information from a PAM provider. See *Privileged Access Management (PAM)* in the *Keyfactor Command Reference Guide* 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.

Name	Description

Name	Description

Value	Description
	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 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.

Name	Description
	Name / **Description** **Value** / **Description** See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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. **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

Name	Description

Name	Description

Value	Description
	approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 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

Name	Description

Name	Description
	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 1: 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-	A string containing the variable that the response from the request will be returned in, if any. You can

Name	Description
	Name / **Description**

Value	Description
erty	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)`
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 Active Directory Service Accounts for Keyfactor Command* in the *Keyfactor Command Server Installation Guide*).

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
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)* in the *Keyfactor Command Reference Guide* 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*. 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)`

Name	Description

Name	Description

Value	Description
	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.
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

Name	Description

Name	Description

Value	Description
	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 Table 1: 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

Name	Description
	Name / **Description** **Value** / **Description** 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)` Verb — A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
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 3).
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 3). 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)* in the *Keyfactor Command Reference Guide* for more information. Due to its sensitive nature, this value is not returned 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 *Gathering Keyfactor Identity Provider Data 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>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>Keyfactor Command Installation in the Keyfactor Command Server Installation Guide).</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:
<code>https://keyfactor.keyexample.com/KeyfactorAPI/Certificates?pq.queryString=CN%20-contains%20%22appsrvr14%22%20AND%20CertStorePath%20-ne%20NULL</code>
Or, with tokens:
<code>https://keyfactor.keyexample.com/KeyfactorAPI/Certificates/$(certid)</code>
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:
<code>192.168.12.0/24,192.168.14.22/24</code>
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.</td></tr>
<tr><td>ContentTy-</td><td>A string indicating the content type for the request.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

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).

Signals	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>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>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:
<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 Workflow Definition Operations: Adding or Modifying a Workflow Definition in the Keyfactor Command Reference Guide for an example).</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>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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.4 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* in the *Keyfactor Command Reference Guide*) 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 above) 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 710: 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 711: 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 *Extension/Scripts* in the *Keyfactor API Reference Guide* 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* in the *Keyfactor Command Reference Guide*. **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 Issued Request Alert Operations in the Keyfactor Command Reference Guide). - 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 </td></tr></table>

Name	Description
	Name / **Description**
	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 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**

<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>Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately 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 <a>SMTP on page 1464) 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 <a>Extensions Scripts on page 870).
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</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Value	Description
	`Platform \ExtensionLibrary\net6.0\Workflow`

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:AppOwnerFirstName)</td></tr>\n<tr><td>DN: $(re-

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	quest: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 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: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 envir-

Name	Description

Name	Description

Value	Description

onments 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 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

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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. • Load the secret information from a PAM provider. See *Privileged Access Management (PAM)* in the *Keyfactor Command Reference Guide* 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.

Name	Description

Name	Description

Value	Description
	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 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.

Name	Description
	Name / **Description**

Name

Description

Value	Description
	See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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. **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

Name	Description
	Name / **Description**

Name

Description

Value	Description
	approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 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

Name	Description

Name	Description
	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 1: 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-	A string containing the variable that the response from the request will be returned in, if any. You can

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
erty	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)`
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 Active Directory Service Accounts for Keyfactor Command* in the *Keyfactor Command Server Installation Guide*).

Name	Description
	Name / **Description**

Name

Description

Value	Description
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)* in the *Keyfactor Command Reference Guide* 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*. 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)`

Name	Description

Name	Description

Value	Description
	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.
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

Name	Description

Name	Description

Value	Description
	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 Table 1: 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

Name	Description
	Name / **Description** **Value** / **Description** 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)` Verb — A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
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 3).
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 3). 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)* in the *Keyfactor Command Reference Guide* for more information. Due to its sensitive nature, this value is not returned 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 *Gathering Keyfactor Identity Provider Data for the*

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Keyfactor Command Installation in the *Keyfactor Command Server Installation Guide*).

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.

<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>pe</td><td>Supported values are:
application/json</td></tr>
<tr><td>RequestC-ontent</td><td>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):

<pre>{
   "Id": "$(certid)",
   "Metadata":{
      "RevocationComment": "$(cmnt)"
   }
}</pre>
Note: This example assumes you have a metadata field called RevocationComment.</td></tr>
</table>

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).
</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:</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>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:
<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 Workflow Definition Operations: Adding or Modifying a Workflow Definition in the Keyfactor Command Reference Guide for an example).</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>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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.5 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 1666) to return details including the workflow steps.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/definitions/read/

Table 712: 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* in the *Keyfactor Command Reference Guide*. 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 713: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.6 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 1754). This method returns HTTP 200 OK on a success with details about the workflow definition.

Tip: The following permissions (see *Security Roles and Claims* in the *Keyfactor Command Reference Guide*) 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 714: 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 1593) to retrieve a list or your certificate templates to determine the template ID. Use the GET /CertificateCollections method (see GET Certificate Collections on page 461) 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 715: 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 *Extension/Scripts* in the *Keyfactor API Reference Guide* 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* in the *Keyfactor Command Reference Guide*. **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 Issued Request Alert Operations in the Keyfactor Command Reference Guide). - 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 </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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 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**</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>Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately 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 <a>SMTP on page 1464) 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 <a>Extensions Scripts on page 870).
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</code></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Value	Description
	`Platform \ExtensionLibrary\net6.0\Workflow`

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:AppOwnerFirstName)</td></tr>\n<tr><td>DN: $(re-

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	quest: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 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: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 envir-

Name	Description

Name	Description

Value	Description

onments 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 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

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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. • Load the secret information from a PAM provider. See *Privileged Access Management (PAM)* in the *Keyfactor Command Reference Guide* 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.

Name	Description

Name	Description

Value	Description
	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 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.

Name	Description

Name	Description

Value	Description
	See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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. **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

Name	Description
	Name / **Description**

Name

Description

Value	Description
	approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 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

Name	Description

Name	Description
	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 1: 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-	A string containing the variable that the response from the request will be returned in, if any. You can

Name	Description
	Name / **Description**

Value	Description
erty	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)`
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 Active Directory Service Accounts for Keyfactor Command* in the *Keyfactor Command Server Installation Guide*).

Name	Description
	Name / **Description**:

Value	Description
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)* in the *Keyfactor Command Reference Guide* 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*. 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)`

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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.
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

Name	Description

Name	Description

Value	Description
	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 Table 1: 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

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	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)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>client_id</td><td>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 3).</td></tr><tr><td>client_secret</td><td>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 3). 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) in the Keyfactor Command Reference Guide for more information. Due to its sensitive nature, this value is not returned in responses.</td></tr><tr><td>TokenEndpoint</td><td>A string container the URL of the token endpoint for your identity provider. For example: <code>https://my-keyidp-server.keyexample.com/realms/Keyfactor/protocol/openid-connect/token</code> 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 Gathering Keyfactor Identity Provider Data for the</td></tr></table></td></tr></table>

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Keyfactor Command Installation in the *Keyfactor Command Server Installation Guide*).

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).

Signals	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:

Name	Description

Name	Description

Value	Description
RoleIds	An array of integers indicating the security roles whose members are allowed to approve the request.
SignalName	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*.

⚠ **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.

Conditions

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:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID of the condition.
Value	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 *Workflow Definition Operations: Adding or Modifying a Workflow Definition* in the *Keyfactor Command Reference Guide* for an example).

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.7 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/definitions/read/

Table 716: 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* in the *Keyfactor Command Reference Guide*. 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 717: 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 *Extension/Scripts* in the *Keyfactor API*

Name	Description
	Reference Guide 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* in the *Keyfactor Command Reference Guide*. **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* in the *Keyfactor Command Reference Guide*). • **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

Name	Description
	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. **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,

Name	Description
	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 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**

Name	Description
	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.
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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.8 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/definitions/read/

Table 718: 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* in the *Keyfactor Command Reference Guide*. 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 719: 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.9 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* in the *Keyfactor Command Reference Guide*) 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 720: 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 1716) to retrieve a list of all the workflow definitions to determine the GUID.

Name	In	Description
request	Body	

Name	Description
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 • 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 *Extension/Scripts* in the *Keyfactor API Reference Guide* 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

Table 721: 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.

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 *Extension/Scripts* in the *Keyfactor API Reference Guide* 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* in the *Keyfactor Command Reference Guide*. **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 Issued Request Alert Operations in the Keyfactor Command Reference Guide). - 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 </td></tr></table>

Name	Description
	Name / **Description**
	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 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**

<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>Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately 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 <a>SMTP on page 1464) 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 <a>Extensions Scripts on page 870).
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</code></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Value	Description
	`Platform \ExtensionLibrary\net6.0\Workflow`

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:AppOwnerFirstName)</td></tr>\n<tr><td>DN: $(re-

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	quest: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 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: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 envir-

Name	Description

Name	Description

Value	Description

onments 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 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

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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. • Load the secret information from a PAM provider. See *Privileged Access Management (PAM)* in the *Keyfactor Command Reference Guide* 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.

Name	Description

Name	Description

Value	Description
	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 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.

Name	Description
	Name / **Description**

Name

Description

Value	Description
	See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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. **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

<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>approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML.
See Table: Tokens for Workflow Definitions in the Keyfactor Command Reference Guide for a complete list of available tokens.</td></tr>
<tr><td>ApprovalE-mailRecipients</td><td>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 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>
</table>
Tip: Tokens (a.k.a. substitutable special text) may be used in
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description
	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 1: 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-	A string containing the variable that the response from the request will be returned in, if any. You can

Name	Description
	Name / **Description**

Value	Description
erty	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)`
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 Active Directory Service Accounts for Keyfactor Command* in the *Keyfactor Command Server Installation Guide*).

Name	Description
	Name / **Description**

Name

Description

Value	Description
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)* in the *Keyfactor Command Reference Guide* 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*. 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)`

Name	Description

Name	Description

Value	Description
	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.
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

Name	Description

Name	Description

Value	Description
	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 Table 1: 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

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	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)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
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 3).
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 3). 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)* in the *Keyfactor Command Reference Guide* for more information. Due to its sensitive nature, this value is not returned 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 *Gathering Keyfactor Identity Provider Data for the*

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Keyfactor Command Installation in the *Keyfactor Command Server Installation Guide*).

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).

Signals	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:

Name	Description

Name	Description

Value	Description
RoleIds	An array of integers indicating the security roles whose members are allowed to approve the request.
SignalName	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*.

⚠ **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.

Conditions

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:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID of the condition.
Value	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 *Workflow Definition Operations: Adding or Modifying a Workflow Definition* in the *Keyfactor Command Reference Guide* for an example).

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.40.10 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 Workflow Instances on page 1805) 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/definitions/modify/

Table 722: 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 1716) to retrieve a list of all the workflow definitions to determine the GUID.

Table 723: 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.

<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 *Extension/Scripts* in the *Keyfactor API Reference Guide* 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* in the *Keyfactor Command Reference Guide*. **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 Issued Request Alert Operations in the Keyfactor Command Reference Guide). - 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 </td></tr></table>

Name	Description
	Name / **Description**
	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 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**

<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>Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately 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 <a>SMTP on page 1464) 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 <a>Extensions Scripts on page 870).
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</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Value	Description
	`Platform \ExtensionLibrary\net6.0\Workflow`

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:AppOwnerFirstName)</td></tr>\n<tr><td>DN: $(re-

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	quest: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 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: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 envir-

Name	Description

Name	Description

Value	Description

onments 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 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

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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. • Load the secret information from a PAM provider. See *Privileged Access Management (PAM)* in the *Keyfactor Command Reference Guide* 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.

Name	Description

Name	Description

Value	Description
	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 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.

Name	Description
	Name / **Description** **Value** / **Description** See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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. **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

Name	Description
	Name / **Description**

Name

Description

Value	Description
	approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* 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 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

Name	Description

Name	Description
	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 1: 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-	A string containing the variable that the response from the request will be returned in, if any. You can

Name	Description

Name	Description

Value	Description
erty	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)`
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 Active Directory Service Accounts for Keyfactor Command* in the *Keyfactor Command Server Installation Guide*).

Name	Description

	Name / **Description**
	Value / **Description**
	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)* in the *Keyfactor Command Reference Guide* 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*.
	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)`

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	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.
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

Name	Description

Name	Description

Value	Description
	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 Table 1: 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

Name	Description
	Name / **Description**

Name	Description
	Value / **Description**

Value	Description
	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)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS • POST • PUT • TRACE

Name	Description
	Name / **Description**

Value	Description
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 3).
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 3). 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)* in the *Keyfactor Command Reference Guide* for more information. Due to its sensitive nature, this value is not returned 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 *Gathering Keyfactor Identity Provider Data for the*

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Keyfactor Command Installation in the *Keyfactor Command Server Installation Guide*).

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).

Signals	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:

Name	Description

Name	Description

Value	Description
RoleIds	An array of integers indicating the security roles whose members are allowed to approve the request.
SignalName	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*.

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.

Conditions

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:

Value	Description
Id	A string indicating the Keyfactor Command reference GUID of the condition.
Value	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 *Workflow Definition Operations: Adding or Modifying a Workflow Definition* in the *Keyfactor Command Reference Guide* for an example).

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.41 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 724: Workflow Instances Endpoints

Endpoint	Method	Description	Link
/{instanceId}	DELETE	Delete the workflow instance with he specified GUID.	DELETE Workflow Instances Instance Id on the next page
/{instanceId}	GET	Retrieve the workflow instance with the specified GUID.	GET Workflow Instances Instance ID on page 1807
/	GET	Retrieve a list of the workflow	GET Workflow Instances

Endpoint	Method	Description	Link
		instances.	on page 1832
/My	GET	Retrieve the workflow instances created by the user making the API request.	GET Workflow Instances My on page 1837
/AssignedToMe	GET	Retrieve the workflow instances assigned to the user making the API request.	GET Workflow Instances AssignedToMe on page 1842
/{instanceId}/Stop	POST	Rejects a workflow instance, preventing it from continuing.	POST Workflow Instances Instance Id Stop on page 1847
/{instanceId}/Signals	POST	Input data to the workflow instance with the specified GUID.	POST Workflow Instances Instance ID Signals on page 1848
/{instanceId}/Restart	POST	Restart the specified workflow instance after a failure.	POST Workflow Instances Instance Id Restart on page 1850

2.6.41.1 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/instances/manage/

Table 725: 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 1832) to retrieve a list of all the workflow instances to determine the GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.41.2 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* in the *Keyfactor Command Reference Guide*) 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 726: 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 1832) 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 727: 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>

Name	Description

Name	Operation Type	Description
OperationStart	Revocation	A string indicating the time at which the revocation workflow was initiated.
Template	Enrollment	A string indicating the certificate template short name used for the enrollment request.
IncludeChain	Enrollment	A Boolean indicating whether to include the certificate chain in the enrollment response (true) or not (false).
SANs	Enrollment	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:

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:

Name	Description

	<table><tr><th>Name</th><th>Operation Type</th><th>Description</th></tr><tr><td></td><td></td><td><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><tr><td>Metadata</td><td>Enrollment</td><td>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.</td></tr><tr><td>Format</td><td>Enrollment</td><td>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.</td></tr><tr><td>CustomName</td><td>Enrollment</td><td>A string indicating a custom friendly name for the certificate.</td></tr><tr><td>Subject</td><td>Enrollment</td><td>A string containing the subject name of the requested certificate using X.500 format.</td></tr><tr><td>RenewalCertificate</td><td>Enrollment</td><td>An object containing the certificate information for the certificate that is being renewed. Certificate data includes:</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>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 Renew in the Keyfactor Command Reference Guide and POST Enrollment Renew on page 861).
</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 492) 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* 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 279) 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>Oper-ation 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>Revoc-ation</td><td>A string containing the date and time when the certificate will be revoked.</td></tr>
<tr><td>Comment</td><td>Revoc-ation</td><td>A string containing a freeform reason or comment on why the certificate is being revoked.</td></tr>
<tr><td>Delegate</td><td>Revoc-ation</td><td>A Boolean indicating whether delegation is enabled for the certificate authority that issued the certi-ficate (true) or not (false).</td></tr>
<tr><td>OperationStart</td><td>Revoc-ation</td><td>A string indicating the time at which the revocation workflow was initiated.</td></tr>
<tr><td>Template</td><td>Enroll-ment</td><td>A string indicating the short certificate template name used for the enrollment request.</td></tr>
<tr><td>IncludeChain</td><td>Enroll-ment</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>Enroll-ment</td><td>An object indicating the subject alternative names (SANs) for the certificate requested in the enroll-ment, 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>Oper-ation 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 Iden-tifier</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_ntprin-cipalname</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>Addi-tionalAttributes</td><td>Enroll-ment</td><td>An object indicating values for any custom enroll-ment 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 Renew in the Keyfactor Command Reference Guide and <a>POST Enrollment Renew on page 861).</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 492) 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 in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>Over-write</td><td>A Boolean that sets whether a certificate in the store with the Alias provided should be overwritten</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Operation Type** / **Description** **Name** / **Description** 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 279) 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 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: `"JobProperties": [ "NetscalerVserver" ]` It can be seen in the Keyfactor

Name	Description

Name	Operation Type	Description

Name	Description

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:

```
"Properties":
{"NetscalerVserver":"My
VirtualServerName"}
```

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.

ManagementJobTime | Enrollment | An object indicating the schedule for the management job to add the certificate to any certificate store(s). Possible management job time values include:

Name	Description
Immediate	A Boolean that indicates a job scheduled to run immediately (true) or not (false).

<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>Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as null.</td></tr>
<tr><td>ExactlyOn-ce</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:m-m: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>

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 certi-ficate 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 indic-ating 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>Dispos-itionMessage</td><td>Enroll-ment</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>CACer-tificateRequest</td><td>Enroll-ment</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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.41.3 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/instances/read/

Table 728: 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* in the *Keyfactor Command Reference Guide*. 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 729: 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>

<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:
<code>"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=appsrvr14.keyexample.com."</code>
Or
<code>"KEYEXAMPLE\\jsmith is revoking certificate with CN=appsrvr12.keyexample.com."</code></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.41.4 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/instances/read/
OR
/workflows/instances/read/mine/

Table 730: 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* in the *Keyfactor Command Reference Guide*. 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 731: 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:
<code>"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=appsrvr14.keyexample.com."</code>
Or
<code>"KEYEXAMPLE\\jsmith is revoking certificate with CN=appsrvr12.keyexample.com."</code></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.41.5 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/instances/read/
OR
/workflows/instances/read/pending/

Table 732: 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* in the *Keyfactor Command Reference Guide*. 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 733: 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:
<code>"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=appsrvr14.keyexample.com."</code>
Or
<code>"KEYEXAMPLE\\jsmith is revoking certificate with CN=appsrvr12.keyexample.com."</code></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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.41.6 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/instances/manage/

Table 734: 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 1832) to retrieve a list of all the workflow instances to determine the GUID.

Tip: See the *Keyfactor API Reference and Utility*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.41.7 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* in the *Keyfactor Command Reference Guide*) 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 735: 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 1832) 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:
<code>RequireApproval1.ApprovalStatus</code>
Use the GET /Workflow/Definitions/{definitionid} method (see <a>GET Workflow Definitions Definition ID on page 1666) 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 1832, <a>GET Workflow Instances AssignedToMe on page 1842, or <a>GET Workflow Instances My on page 1837) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.6.41.8 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* in the *Keyfactor Command Reference Guide*) are required to use this feature:
/workflows/instances/manage/
/workflows/instances/read/

Table 736: 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 1832) 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*Keyfactor API Reference and Utility which provides a utility through which the Keyfactor API endpoints can be called and results

returned. It is intended to be used primarily for validation, testing and workflow development. It also serves secondarily as documentation for the API. The link to the Keyfactor API Reference and Utility is in the dropdown from the help icon (?) at the top of the Management Portal page next to the **Log Out** button.

2.7 API Change Log

In this section you will find the change history for the Keyfactor API endpoints from version 9.0 onwards.

2.7.1 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 1853
Incremental	October 2021	API Change Log v9.2 on page 1854
Incremental	November 2021	API Change Log v9.3 on page 1855
Incremental	December 2021	API Change Log v9.4 on page 1855
Incremental	January 2022	API Change Log v9.5 on page 1855
Incremental	February 2022	API Change Log v9.6 on page 1856
Incremental	March 2022	API Change Log v9.7 on page 1856
Incremental	April 2022	API Change Log v9.8 on page 1856
Incremental	May 2022	API Change Log v9.9 on page 1856

2.7.1.1 API Change Log v9.0

API changes for Keyfactor Command version 9.0 Major release

Table 737: 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	

2.7.1.2 API Change Log v9.1

API changes for Keyfactor Command version 9.1 incremental release

Table 738: 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	

2.7.1.3 API Change Log v9.2

API changes for Keyfactor Command version 9.2 incremental release

Table 739: 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.

2.7.1.4 API Change Log v9.3

API changes for Keyfactor Command version 9.3 incremental release

Table 740: API Change Log v9.3

Endpoint	Methods	Action	Notes
/JobTypes/Custom	POST	Fix	No longer requires default field values.

2.7.1.5 API Change Log v9.4

API changes for Keyfactor Command version 9.4 incremental release

Table 741: API Change Log v9.4

Endpoint	Methods	Action	Notes
/Workflow/Certificates/Pending	GET	Update	Now returns the associated metadata.

2.7.1.6 API Change Log v9.5

API changes for Keyfactor Command version 9.5 incremental release

Table 742: API Change Log v9.5

Endpoint	Methods	Action	Notes
/Enrollment/PFX	POST	Update	No longer requires a certificate authority name to be provided.

2.7.1.7 API Change Log v9.6

API changes for Keyfactor Command version 9.6 incremental release.

No API endpoint changes were made in this release.

2.7.1.8 API Change Log v9.7

API changes for Keyfactor Command version 9.7 incremental release

Table 743: API Change Log v9.7

Endpoint	Methods	Action	Notes
/KeyfactorAPI/License	GET	Add	

2.7.1.9 API Change Log v9.8

API changes for Keyfactor Command version 9.8 incremental release.

No API endpoint changes were made in this release.

2.7.1.10 API Change Log v9.9

API changes for Keyfactor Command version 9.9 incremental release

Table 744: 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.

2.7.2 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 1863
Incremental	January 2023	API Change Log v10.2 on page 1863
Hot Fix	April 2023	API Change Log v10.3.1 on page 1863
Incremental	May 2023	API Change Log v10.4 on page 1864
Hot Fix	July 2023	API Change Log v10.4.3 on page 1864
Hot Fix	September 2023	API Change Log v10.4.5 on page 1865
Hot Fix	September 2023	API Change Log v10.4.6 on page 1865

2.7.2.1 API Change Log v10.0

API changes for Keyfactor Command version 10.0 Major release

Table 745: 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	

2.7.2.2 API Change Log v10.1

API changes for Keyfactor Command version 10.1 incremental release

Table 746: 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.

2.7.2.3 API Change Log v10.2

API changes for Keyfactor Command version 10.2 incremental release

Table 747: 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.

2.7.2.4 API Change Log v10.3.1

API changes for Keyfactor Command version 10.3.1 hot fix release

Table 748: 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.

2.7.2.5 API Change Log v10.4

API changes for Keyfactor Command version 10.4 incremental release

Table 749: 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.

2.7.2.6 API Change Log v10.4.3

API changes for Keyfactor Command version 10.4.3 hot fix release

Table 750: 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.

2.7.2.7 API Change Log v10.4.5

API changes for Keyfactor Command version 10.4.5 hot fix release

Table 751: 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.

2.7.2.8 API Change Log v10.4.6

API changes for Keyfactor Command version 10.4.6 hot fix release

Table 752: 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.

2.7.3 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

2.7.3.1 API Change Log v11.0

API changes for Keyfactor Command version 11.0 major release

Table 753: 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.

3.0 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.

4.0 Copyright Notice