Keyfactor Web APIs 10.2

Reference Guide

Table of Contents

List of Figures

List of Tables

1.0 Introduction

The *Keyfactor Command Documentation Suite* includes:

- *Keyfactor Command Reference Guide*
- *Keyfactor Web APIs Reference Guide*
- *Keyfactor Command Server Installation Guide*
- *Keyfactor Orchestrators Installation and Configuration Guide*
- *Keyfactor Command Release Notes*

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 Web APIs Reference

The Keyfactor Command solution by Keyfactor exposes Web APIs 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 APIs complement the web components of Keyfactor Command and offer a number of HTTP method calls that provide similar functionality to that available within the portal's user interface, but which can be accessed programmatically by any system capable of making web requests. These APIs have 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.

2.1 Overview

Keyfactor exposes two APIs for external use:

- The Keyfactor API was introduced in Keyfactor Command version 6.1 and is the newer API. Customers should be using this API going forward.
- The Classic API has been provided in the product for several product generations and is continuing to be supported for legacy implementations but should not be used for new implementations.

2.1.1 Transaction Security

The Keyfactor Web APIs rely on SSL/TLS to protect the HTTP communications between the client and Keyfactor Command server. In a typical deployment, the APIs 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 these APIs. SSL is used to protect the confidentiality of user credentials; therefore, SSL should be used with the Keyfactor Web APIs.

Keyfactor recommends that any device using these APIs 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.

There is no longer the need to configure an API application with a key and secret and a particular template in the portal to allow for enrollment for a certificate with the API. Certificate enrollment no longer requires a key and secret and enrollment permissions are now controlled on the template level.

Finally, access to the API methods can be limited per client to a maximum request frequency. The amount of time required between calls can also be configured in the Keyfactor Command Management Portal Application Settings for the APIs. 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.

2.1.2 Architecture

By default, all Web API methods start with a base path, which varies depending on the API and corresponds to an application under IIS; this path is configurable at install time. For the Keyfactor API, the default base path is *KeyfactorApi*. The API component name, version number (only applicable to the Classic API), and method name then comprise the second through fourth 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 in the Keyfactor API and "/CMSApi/CertEnroll/1/Token" would be the URL format for the Token method of version 1 of the CertEnroll API component in the Classic API. Version numbers are only used in the URL for the Classic API.

2.1.3 Web API Common Features

Some aspects of the Web 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.

Additionally, many Classic API methods operate on a certificate resource stored in Keyfactor Command, and a standardized way to identify the certificate for the operation is used in the request structure across several Classic API components; this is described in Table 4: Classic API Certificate Lookup Structure. This table does not apply to the Keyfactor API.

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; charset=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 generated by most

Header Name	API Version	Header Value	Description
			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	API Version	Header Value	Description
Cache-Control	Both	no-cache	API requests are generally not cacheable. Note that this is not respected by all client systems.
Pragma	Both	no-cache	API requests are generally not cacheable. Note that this is not respected by all client systems.
Content-Length	Both	<varies>	Length of the HTTP response.
Content-Type	Both	application/json	Most calls return application/json, but occasionally text/-plain or text/xml.
Expires	Both	-1	Usually ignored.
Server	Both	<varies>	Software version reported by IIS platform hosting Keyfactor Command.
X-CSS-CMS-APIVer-sion	Classic API	2.0	Classic API version accessed (see usage in Versioning on page 6).
X-CSS-CMS-CMSVer-sion	Classic API	10.2	Keyfactor Command platform version.
X-Keyfactor-Product-Version	Keyfactor API	<varies>	Keyfactor Command platform version.
X-Total-Count	Keyfactor API	<varies>	Total number of elements returned.
X-AspNet-Version	Both	<varies>	Version of ASP.NET supporting Keyfactor Command installation.
X-Powered-By	Both	ASP.NET	Header added by underlying ASP.NET implementation.

Header Name	API Version	Header Value	Description
Date	Both	<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.

Table 4: Classic API Certificate Lookup Structure

Parameter Name	Parameter Value
Type	One of "Serial", "Thumbprint", and "CMSID".
SerialNumber	Hexadecimal serial number of referenced certificate. Required only if Type is "Serial".
IssuerDN	Distinguished Name of the issuer of the referenced certificate. Required only if Type is "Serial".
Thumbprint	SHA-1 thumbprint of the referenced certificate. Required only if Type is "Thumbprint".
CMSID	Identifier assigned by Keyfactor Command to the referenced certificate. Required only if Type is "CMSID".

2.1.4 Versioning

The Keyfactor Web APIs are versioned as a set and released in conjunction with Keyfactor Command at the same version level (e.g. version 10.2). In addition, both the Keyfactor API and the Classic API[1] 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.

The Classic API provides various methods to retrieve the version of Keyfactor Command. For example, values for both the Classic API version and the Keyfactor Command version are returned in HTTP headers with each response to an API call. Additionally, the *Status* endpoint (see Status on page 1479) provides additional information about the capabilities of the Classic API in its installed version. The Keyfactor API does not presently have an equivalent functionality.

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.

Several endpoints of the Classic API have their own incremental versioning. For example, the CertEnroll endpoint has three versions, the most recent of which is three:

- CertEnroll/1
- CertEnroll/2
- CertEnroll/3

As the Keyfactor Web APIs have evolved and continue to evolve, an additional security constraint is available to limit access to deprecated legacy versions of API endpoints. In many cases, newer versions of an endpoint are more secure and robust, easier to use, and offer more functionality. Keyfactor highly recommends use of the newest endpoints wherever possible. To this end, it is possible to disable deprecated API endpoints in the Classic API from the API Application Settings within the Keyfactor Command Management Portal. In Keyfactor Command 10.2, this setting will disable the following endpoints:

- CertEnroll/1/Token[2]
- CertEnroll/1/Status[1]
- CertEnroll/1/Certificates/Pkcs10[1]
- CertEnroll/1/Certificates/Pkcs12[1]

---

[1]The Classic API was historically versioned on a different release schedule to Keyfactor Command and so has separate reporting of versions for itself and Keyfactor Command.
[2]The CertEnroll v1 endpoints are deprecated.

- Metadata/2/Set
- Metadata/2/Get
- Metadata/2/Compare
- Certificates/1/Metafield
- Certificates/1/Import[1]

If the *Allow Deprecated API Calls* setting is disabled, any client attempting to access one of these endpoints will receive an error message instead of the expected results. This will, of course, prevent client applications that rely on these endpoints from functioning, and if these applications cannot be updated to the newer endpoints then the *Allow Deprecate API Calls* setting must be enabled. Otherwise, Keyfactor recommends that these endpoints be disabled to reduce exposure to unauthorized or unintended use.

The following endpoints have been removed from the Classic API and are no longer supported:

- CertEnroll/1/Templates

Note: API versioning strategy in Keyfactor Command shifted somewhat between versions 4.0 and 5.0 (when the product was known as Certificate Management System or CMS). As such, the API versioning mechanisms described in CMS 4.0-4.5 documentation, while still generally correct, are no longer our primary recommendation.

2.2 Keyfactor API

The Keyfactor API is the Web API introduced in Keyfactor Command version 6.1. It is designed to support the updated platform architecture in the new version of the main Keyfactor Command solution and to, in time, replace the Classic 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 Web APIs 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 Endpoint 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 Endpoint Utility*.

[1]The Certificates/1/Import endpoint, using a multipart/form-data request, is no longer supported by Keyfactor for customers that are not currently using it.

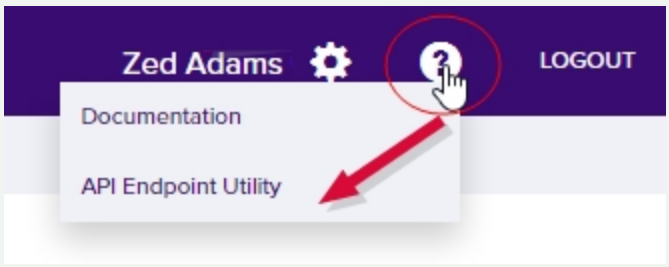


Figure 1: Documentation in the Help Dropdown

You can also browse to the *Keyfactor API Endpoint 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#
```

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.

A static reference (without the interactive utility you can find in the Keyfactor Command Management Portal) is available as a zip file in the Keyfactor Client Portal[1].

2.2.1 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 5: 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 12
/Reset	POST	Resets one or more orchestrators or agents to a new state and clears jobs.	POST Agents Reset on page 16
/Approve	POST	Approves an orchestrator.	POST Agents Approve on

[1]Embedded links to external documents point to the document on the Keyfactor Client Portal. Access to the portal requires a login. See your administrator or your Keyfactor Client Success representative to obtain a login to the portal.

Endpoint	Method	Description	Link
			page 17
/Disapprove	POST	Disapproves an orchestrator.	POST Agents Disapprove on page 17
/{id}/Reset	POST	Resets a single orchestrator or agent to a new state and clears jobs.	POST Agents ID Reset on page 18
/{id}/FetchLogs	POST	Schedules a job on the orchestrator or agent to retrieve log files.	POST Agents ID FetchLogs on page 19
/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 19

2.2.1.1 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 Overview) are required to use this feature:
AgentManagement: *Read*

Table 6: 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 12) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Table 7: 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>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Capabilities</td><td>An array of strings indicating the capabilities reported by the orchestrator. These may be built-in or custom capabilities. Possible built-in values for common orchestrators include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>AWS</td><td>Amazon Web Services</td></tr>
<tr><td>NS</td><td>NetScaler</td></tr>
<tr><td>F5-CA-REST</td><td>F5 CA Bundles (REST)</td></tr>
<tr><td>F5-WS-REST</td><td>F5 Web Server (REST)</td></tr>
<tr><td>F5-SL-REST</td><td>F5 SSL Profile (REST)</td></tr>
<tr><td>IIS</td><td>IIS</td></tr>
<tr><td>FTP</td><td>File Transfer Protocol</td></tr>
<tr><td>F5</td><td>F5 SSL Profile and F5 Web Server (SOAP)</td></tr>
<tr><td>CA</td><td>Remote CA Management</td></tr>
<tr><td>SSL</td><td>SSL Discovery and Monitoring</td></tr>
<tr><td>MacEnrollment</td><td>Mac Autoenrollment</td></tr>
<tr><td>JKS</td><td>Java Keystore</td></tr>
<tr><td>PEM</td><td>PEM Store</td></tr>
<tr><td>LOGS</td><td>Fetch Logs</td></tr>
<tr><td>TemplateSync</td><td>Template Synchronization</td></tr>
</table></td></tr>
<tr><td>Blueprint</td><td>A string indicating the name of the blueprint associated with the orchestrator.</td></tr>
<tr><td>Thumbprint</td><td>A string indicating the thumbprint of the certificate that Keyfactor Command is expecting the orchestrator to use for client certificate authentication.</td></tr>
<tr><td>LegacyThumbprint</td><td>A string indicating the thumbprint of the certificate previously used by the orchestrator for client certificate authentication before a certificate renewal operation took place (rotating the current thumbprint into the legacy thumbprint). The legacy thumbprint is cleared once the orchestrator successfully registers with the new thumbprint.</td></tr>
<tr><td>AuthCertificateReenrollment</td><td>An integer indicating the value of the orchestrator certificate reenrollment request or require status. Possible values are:</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—Unset the value so that the orchestrator will not request a new client authentication certificate (based on this value).</td></tr>
<tr><td>1</td><td>Requested—The orchestrator will request a new client authentication certificate when it next registers for a session. Orchestrator activity will be allowed to continue as usual.</td></tr>
<tr><td>2</td><td>Required—The orchestrator will request a new client authentication certificate when it next registers for a session. A new session will not be granted and orchestrator activity will not be allowed to continue until the orchestrator acquires a new certificate.</td></tr>
</table>
</td></tr>
<tr><td>LastThumbprintUsed</td><td>A string indicating the thumbprint of the certificate that the orchestrator most recently used for client certificate authentication. In most cases, this will match the Thumbprint.</td></tr>
<tr><td>LastErrorCode</td><td>An integer indicating the last error code, if any, reported from the orchestrator when trying to register a session. This code is cleared on successful session registration.</td></tr>
<tr><td>LastErrorMessage</td><td>A string indicating the last error message, if any, reported from the orchestrator when trying to register a session. This message is cleared on successful session registration.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.1.2 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 Overview) are required to use this feature:
AgentManagement: *Read*

Table 8: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Orchestrator Management Search Feature*. The query fields supported for this endpoint are: • *AgentId* • *Blueprint* • *Capabilities* (See Table 9: 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.
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 9: 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>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Capabilities</td><td>An array of strings indicating the capabilities reported by the orchestrator. These may be built-in or custom capabilities. Possible built-in values for common orchestrators include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>AWS</td><td>Amazon Web Services</td></tr>
<tr><td>NS</td><td>NetScaler</td></tr>
<tr><td>F5-CA-REST</td><td>F5 CA Bundles (REST)</td></tr>
<tr><td>F5-WS-REST</td><td>F5 Web Server (REST)</td></tr>
<tr><td>F5-SL-REST</td><td>F5 SSL Profile (REST)</td></tr>
<tr><td>IIS</td><td>IIS</td></tr>
<tr><td>FTP</td><td>File Transfer Protocol</td></tr>
<tr><td>F5</td><td>F5 SSL Profile and F5 Web Server (SOAP)</td></tr>
<tr><td>CA</td><td>Remote CA Management</td></tr>
<tr><td>SSL</td><td>SSL Discovery and Monitoring</td></tr>
<tr><td>MacEnrollment</td><td>Mac Autoenrollment</td></tr>
<tr><td>JKS</td><td>Java Keystore</td></tr>
<tr><td>PEM</td><td>PEM Store</td></tr>
<tr><td>LOGS</td><td>Fetch Logs</td></tr>
<tr><td>TemplateSync</td><td>Template Synchronization</td></tr>
</table></td></tr>
<tr><td>Blueprint</td><td>A string indicating the name of the blueprint associated with the orchestrator.</td></tr>
<tr><td>Thumbprint</td><td>A string indicating the thumbprint of the certificate that Keyfactor Command is expecting the orchestrator to use for client certificate authentication.</td></tr>
<tr><td>LegacyThumbprint</td><td>A string indicating the thumbprint of the certificate previously used by the orchestrator for client certificate authentication before a certificate renewal operation took place (rotating the current thumbprint into the legacy thumbprint). The legacy thumbprint is cleared once the orchestrator successfully registers with the new thumbprint.</td></tr>
<tr><td>AuthCertificateReenrollment</td><td>An integer indicating the value of the orchestrator certificate reenrollment request or require status. Possible values are:</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—Unset the value so that the orchestrator will not request a new client authentication certificate (based on this value).</td></tr>
<tr><td>1</td><td>Requested—The orchestrator will request a new client authentication certificate when it next registers for a session. Orchestrator activity will be allowed to continue as usual.</td></tr>
<tr><td>2</td><td>Required—The orchestrator will request a new client authentication certificate when it next registers for a session. A new session will not be granted and orchestrator activity will not be allowed to continue until the orchestrator acquires a new certificate.</td></tr>
</table>
</td></tr>
<tr><td>LastThumbprintUsed</td><td>A string indicating the thumbprint of the certificate that the orchestrator most recently used for client certificate authentication. In most cases, this will match the Thumbprint.</td></tr>
<tr><td>LastErrorCode</td><td>An integer indicating the last error code, if any, reported from the orchestrator when trying to register a session. This code is cleared on successful session registration.</td></tr>
<tr><td>LastErrorMessage</td><td>A string indicating the last error message, if any, reported from the orchestrator when trying to register a session. This message is cleared on successful session registration.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.1.3 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 Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *Modify*

Table 10: POST Agents Reset Input Parameters

Name	In	Description
agentIds	Body	**Required**. An array of GUIDs of the orchestrators to reset. Use the *GET /Agents* method (see GET Agents on page 12) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.1.4 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 Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *Modify*

Table 11: 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 12) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs and current status of the orchestrators.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.1.5 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 Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *Modify*

Table 12: 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 12) to retrieve a list of all the orchestrators to determine the orchestrator GUIDs and current status of the orchestrators.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.1.6 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 Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *Modify*

Table 13: 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 12) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.1.7 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 Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *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 709).

Table 14: 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 12) to retrieve a list of all the orchestrators to determine the orchestrator GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.1.8 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 details

Tip: The following permissions (see Security Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *Modify*

Table 15: 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 12) 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 16: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.2 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 17: Agent BluePrint Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the orchestrator blueprint with the specified GUID.	DELETE Agent 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 23
/{id}/Jobs	GET	Returns details of the certificate store scheduled	GET Agent BluePrint

Endpoint	Method	Description	Link
		jobs for the orchestrator blueprint with the specified GUID.	ID Jobs on page 24
/{id}/Stores	GET	Returns details of the certificate stores for the orchestrator blueprint with the specified GUID.	GET Agent BluePrint ID Stores on page 28
/ApplyBlueprint	POST	Applies an orchestrator blueprint to one or more orchestrators.	POST AgentBluePrint ApplyBluePrint on page 30
/GenerateBlueprint	POST	Creates a new orchestrator blueprint from an orchestrator.	POST AgentBluePrint GenerateBluePrint on page 31

2.2.2.1 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 Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *Modify*

Table 18: DELETE AgentBluePrint {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 the next page) to retrieve a list of all the blueprints to determine the orchestrator blueprint GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.2.2 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 the next page) or GET /AgentBluePrint/{id}/Stores method (see GET Agent BluePrint ID Stores on page 28).

Tip: The following permissions (see Security Overview) are required to use this feature: AgentManagement: *Read*

Table 19: GET AgentBluePrint {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 below) to retrieve a list of all the blueprints to determine the orchestrator blueprint GUID.

Table 20: GET AgentBluePrint {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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.2.3 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 Overview) are required to use this feature: AgentManagement: *Read*

Table 21: GET AgentBluePrint 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 22: GET AgentBluePrint 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.2.4 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 Overview) are required to use this feature:
AgentManagement: *Read*

Table 23: GET AgentBluePrint {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 23) 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.
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 24: GET AgentBluePrint {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 field is populated only for management jobs.
KeyfactorSchedule	The schedule for the certificate store job. This field is populated only for inventory and discovery jobs.

Name	Description
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 array 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.
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.

Name	Description
	<table><tr><th>Name</th><th>Description</th></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 496).</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. (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)</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><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>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 526 for more information).</td></tr></table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.2.5 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 Overview) are required to use this feature: AgentManagement: *Read*

Table 25: GET AgentBluePrint {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 23) 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.
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 26: GET AgentBluePrint {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 496).
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. (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)
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 526 for more information).

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.2.6 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 Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *Modify*

Table 27: POST AgentBluePrint Apply Input Parameters

Name	In	Description
agentIds	Body	**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 12) 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.
templateId	Body	A string indicating the GUID of the blueprint to apply to the orchestrator(s). Use the *GET AgentBluePrint* method (see GET Agent BluePrint on page 23) to retrieve a list of all the blueprints to determine the blueprint GUIDs.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.2.7 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 Overview) are required to use this feature:
AgentManagement: *Read*
AgentManagement: *Modify*

Table 28: POST AgentBluePrint 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 12) 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 29: POST AgentBluePrint 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).
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).

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.3 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 30: Agent Pool Endpoints

Endpoint	Method	Description	Links
/{id}	DELETE	Deletes the specified orchestrator pool.	DELETE Agent Pools ID on the next page
/{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 35
/	POST	Creates an orchestrator pool based on information in the request.	POST Agent Pools on page 37
/	PUT	Updates an orchestrator pool based on information in the request.	PUT Agent Pools on page 39
/Agents	GET	Returns a list of orchestrators associated with the Default Agent Pool.	GET Agent Pools Agents on page 41

2.2.3.1 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 Overview) are required to use this feature:
SslManagement: *Read*
SslManagement: *Modify*

Table 31: DELETE AgentPools {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 35) to retrieve a list of all the orchestrator pools to determine the orchestrator pool GUID. The Default Agent Pool cannot be deleted.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.3.2 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 32: GET AgentPools {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 35) to retrieve a list of all the orchestrator pools to determine the orchestrator pool GUID.

Table 33: 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.3.3 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 34: GET AgentPools 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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.
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 35: 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.3.4 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 Overview) are required to use this feature:
SslManagement: *Read*
SslManagement: *Modify*

Table 36: POST AgentPools 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 orchestrator pool.</td></tr>
<tr><td>Agents</td><td>Body</td><td>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:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>AgentId</td><td>Required. A string indicating the GUID of the orchestrator being assigned.</td></tr>
<tr><td>EnableDiscover</td><td>Required*. A Boolean that sets whether a discovery job can be sent to this orchestrator. One of EnabledDiscover or EnableMonitor is required.</td></tr>
<tr><td>EnableMonitor</td><td>Required*. A Boolean that sets whether a monitoring job can be sent to this orchestrator. One of EnabledDiscover or EnableMonitor is required.</td></tr>
</table>
</td></tr>
</table>

Table 37: POST 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.3.5 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 Overview) are required to use this feature:
SslManagement: *Read*
SslManagement: *Modify*

Warning: 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 38: PUT AgentPools 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 39: PUT 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.3.6 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 Overview) are required to use this feature: SslManagement: *Read*

Table 40: GET AgentPools 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Collection Manager*. 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.
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 41: GET AgentPools 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.4 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 67
- Alerts Issued on page 102
- Alerts Key Rotation on page 132
- Alerts Pending on page 161

2.2.4.1 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 42: 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 below
/Alerts/Denied	PUT	Updates a denied certificate request alert for the specified ID.	PUT Alerts Denied on page 59
/Alerts/Denied	GET	Retrieves details for all configured denied certificate request alerts.	GET Alerts Denied on page 47
/Alerts/Denied	POST	Creates a new denied certificate request alert.	POST Alerts Denied on page 51

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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 43: 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 47) to retrieve a list of all the issued request alerts to determine the alert ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 44: 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 47) to retrieve a list of all the issued request alerts to determine the alert ID.

Table 45: 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 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}.

<table>
<tr><th>Name</th><th>Description</th></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>
<tr><td>RegisteredEventHandl-er</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>EventHand-lerParameters</td><td>An array 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>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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>
</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature: WorkflowManagement: *Read*

Table 46: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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.
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 47: 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 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}.

<table>
<tr><th>Name</th><th>Description</th></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>
<tr><td>RegisteredEventHandl-er</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>EventHand-lerParameters</td><td>An array 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	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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 48: 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: {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	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. • 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

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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 1206) 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>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 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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: <pre>"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"</pre>

Name	In	Description
		`},{"Id": 31,"Key": "DenialComment","DefaultValue": "cmnt","ParameterType": "Token"},{"Id": 32,"Key": "ScriptName","DefaultValue": "MyScript.ps1","ParameterType": "Script"}]`

Table 49: 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 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}.

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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>
</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Warning: 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 50: 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.
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. • Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>Body</td><td>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 1206) 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>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 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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: <pre>"EventHandlerParameters": [{ "Id": 28, "Key": "cn", "DefaultValue": "rcn", "ParameterType": "Token" }, { "Id": 29, "Key": "AppOwnerFirstName", "DefaultValue": "metadata:AppOwnerFirstName", "ParameterType": "Token" }, { "Id": 30, "Key": "Text",</pre>

Name	In	Description
		"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 51: 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 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}.

<table>
<tr><th>Name</th><th>Description</th></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>
<tr><td>RegisteredEventHandl-er</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>EventHand-lerParameters</td><td>An array 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	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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.4.2 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 52: Alerts Expiration

Endpoint	Method	Description	Link
/Alerts/Expiration/{id}	DELETE	Deletes an expired certificate for the specified ID.	DELETE Alerts Expiration ID below
/Alerts/Expiration/{id}	GET	Retrieves details for an expired certificate 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 73
/Alerts/Expiration/Schedule	PUT	Updates the schedule for delivery of expired certificate alerts.	PUT Alerts Expiration Schedule on page 73
/Alerts/Expiration	GET	Retrieves details for all configured expired certificate.	GET Alerts Expiration on page 75
/Alerts/Expiration	POST	Creates a new expired certificate alert.	POST Alerts Expiration on page 80
/Alerts/Expiration	PUT	Updates an expired certificate for the specified ID.	PUT Alerts Expiration on page 89
/Alerts/Expiration/Test	POST	Test an Expiration Alert	POST Alerts Expiration Test on page 98
/Alerts/Expiration/TestAll	POST	Test All Expiration Alerts	POST Alerts Expiration Test All on page 100

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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 53: 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 75) to retrieve a list of all the expiration alerts to determine the alert ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 54: 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 75) to retrieve a list of all the expiration alerts to determine the alert ID.

Table 55: 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:oo 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:oo am UTC would be alerted on.
Recipients	An object 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>Description</th></tr>
<tr><td></td><td>Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQuery</td><td>An array 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>
<tr><td>RegisteredEventHandler</td><td>An array 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 object 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	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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Web API Common Features on page 3).

Tip: The following permissions (see Security Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 56: GET Alerts Expiration Schedule Response Data

Name	Description
Schedule	An array 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. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }`

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 57: PUT Alerts Expiration Schedule Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Schedule</td><td>Body</td><td>An array indicating the schedule for delivery of the expiration alerts. Possible values are:
<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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table></td></tr>
</table>

Table 58: PUT Alerts Expiration Schedule Response Data

Name	Description
Schedule	An array indicating the schedule for delivery of the expiration alerts. Possible values are: **Name** / **Description** Daily — A dictionary that indicates a job scheduled to run every day at the same time with the parameter: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }`

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 59: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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.
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 60: 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:oo 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:oo am UTC would be alerted on.
Recipients	An object 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>Description</th></tr>
<tr><td></td><td>Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQuery</td><td>An array 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>
<tr><td>RegisteredEventHandler</td><td>An array 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 object 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	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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 61: 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.
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 execu-

Name	In	Description
		tion 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:oo 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:oo am UTC would be alerted on.
Recipients	Body	An object 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}.
CertificateQueryId	Body	**Required**. An integer indicating the certificate collection on which to base the alert. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 363) to retrieve a list of all the certificate collections to determine the collection ID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An array 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 object containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler and is used to reference the fully</td></tr>
</table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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> 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"</pre>

Name	In	Description
		`},{"Id": 30,"Key": "Text","DefaultValue": "Expiration Alert: Enterprise Web Server","ParameterType": "Value"},{"Id": 32,"Key": "ScriptName","DefaultValue": "MyScript.ps1","ParameterType": "Script"}]`

Table 62: 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:oo 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:oo am UTC would be alerted on.
Recipients	An object 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>Description</th></tr>
<tr><td></td><td>Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQuery</td><td>An array 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>
<tr><td>RegisteredEventHandler</td><td>An array 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 object 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	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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature: WorkflowManagement: *Modify*

Warning: 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 63: 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.
ExpirationWarningDays	Body	**Required**. An integer indicating the number of days prior to expiration to send the warning.

Name	In	Description
		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:oo 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:oo am UTC would be alerted on.
Recipients	Body	An object 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}.
CertificateQueryId	Body	**Required**. An integer indicating the certificate collection on which to base the alert. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 363) to retrieve a list of all the certificate collections to determine the collection ID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An array 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 object containing the parameters configured for use by the event handler. The type of data will vary depending on the configured handler. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID of the configured parameter.</td></tr>
<tr><td>Key</td><td>A string indicating the reference name of the configured parameter.</td></tr>
<tr><td>DefaultValue</td><td>A string indicating the value for the parameter. This value is related to the type of parameter (see ParameterType).</td></tr>
<tr><td>ParameterType</td><td>A string containing the parameter type. Supported types are:
LogTarget
This type is used for the event logging handler and is used to reference the fully</td></tr>
</table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>qualified domain name of the target machine to which event should be logged. • Script This type is used for the PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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> 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"</pre>

Name	In	Description
		`},{"Id": 30,"Key": "Text","DefaultValue": "Expiration Alert: Enterprise Web Server","ParameterType": "Value"},{"Id": 32,"Key": "ScriptName","DefaultValue": "MyScript.ps1","ParameterType": "Script"}]`

Table 64: 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:oo 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:oo am UTC would be alerted on.
Recipients	An object 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>Description</th></tr>
<tr><td></td><td>Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>CertificateQuery</td><td>An array 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>
<tr><td>RegisteredEventHandler</td><td>An array 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 object 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	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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*
WorkflowManagement: *Test*

Table 65: POST Alerts Expiration Test Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>expirationAlertTestRequest</td><td>Body</td><td>Required. An array containing information for the alert test. Alert test detail values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>AlertId</td><td>Required. An integer indicating the reference ID of expiration alert to test.
Use the GET /Alerts/Expiration method (see GET Alerts Expiration on page 75) to retrieve a list of all your expiration alerts to determine the alert Id.</td></tr>
<tr><td>EvaluationDate</td><td>Required. A string indicating the start date/time for the test, in UTC.
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.</td></tr>
<tr><td>PreviousEvaluationDate</td><td>Required. A string indicating the end date/time for the test, in UTC.</td></tr>
<tr><td>SendAlerts</td><td>A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is false.</td></tr>
</table>
For example:
<pre>
{
   "EvaluationDate": "2022-08-31T20:51:33.528Z",
   "PreviousEvaluationDate": "2022-08-31T20:51:33.528Z",
   "SendAlerts": true
}
</pre></td></tr>
</table>

Table 66: 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 object 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*
WorkflowManagement: *Test*

Table 67: POST Alerts Expiration Test All Input Parameters

Name	In	Description
expirationAlertTestRequest	Body	**Required**. An array containing information for the alert test. Alert test detail values are: **Name** / **Description** EvaluationDate — **Required**. A string indicating the start date/time for the test, in UTC. 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 — **Required**. A string indicating the end date/time for the test, in UTC. SendAlerts — A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is *false*. For example: `{ "EvaluationDate": "2022-08-31T20:51:33.528Z", "PreviousEvaluationDate": "2022-08-31T20:51:33.528Z", "SendAlerts": true }`

Table 68: 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 object 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.4.3 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 69: 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 108
/Alerts/Issued/Schedule	PUT	Updates the schedule for delivery of issued certificate request alerts.	PUT Alerts Issued Schedule on page 110
/Alerts/Issued	GET	Retrieves details for all configured issued certificate request alerts.	GET Alerts Issued on page 112
/Alerts/Issued	POST	Creates a new issued certificate request alert.	POST Alerts Issued on page 116
/Alerts/Issued	PUT	Updates an issued certificate request alert for the specified ID.	PUT Alerts Issued on page 124

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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 70: 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 112) to retrieve a list of all the issued request alerts to determine the alert ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature: WorkflowManagement: *Read*

Table 71: 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 112) to retrieve a list of all the issued request alerts to determine the alert ID.

Table 72: 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{dnld-link}\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:Ap-pOwnerLastName}</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 Crit-ical: {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. Avail-able 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}.

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EventHandlerParameters</td><td>An array 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Web API Common Features on page 3).

Tip: The following permissions (see Security Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 73: GET Alerts Issued Schedule Response Data

Name	Description
Schedule	An array 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. 2021-05-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
```

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature: WorkflowManagement: *Modify*

Table 74: PUT Alerts Issued Schedule Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Schedule</td><td>Body</td><td>An array indicating the schedule for delivery of the issued request alerts. Possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>
"Interval": {
   "Minutes": 60
}
</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
</pre></td></tr>
</table>
</td></tr>
</table>

Table 75: PUT Alerts Issued Schedule Response Data

Name	Description
Schedule	An array 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. 2021-05-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
```

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 76: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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.
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 77: 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{dnld-link}\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:Ap-pOwnerLastName}</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 Crit-ical: {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. Avail-able 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}.

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EventHandlerParameters</td><td>An array 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 78: 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.
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}

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
• Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>TemplateId</td><td>Body</td><td>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 1206) 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>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 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></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>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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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:
</td></tr>
</table>

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 79: 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{dnld-link}\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:Ap-pOwnerLastName}</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 Crit-ical: {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. Avail-able 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}.

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EventHandlerParameters</td><td>An array 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Warning: 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 80: 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.
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

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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>TemplateId</td><td>Body</td><td>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 1206) 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>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 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></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>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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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:
</td></tr>
</table>

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 81: 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{dnld-link}\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:Ap-pOwnerLastName}</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 Crit-ical: {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. Avail-able 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}.

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EventHandlerParameters</td><td>An array 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.4.4 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 82: Alerts Key Rotation

Endpoint	Method	Description	Link
/Alerts/KeyRotation/{id}	DELETE	Deletes an SSH key rotation alert for the specified ID.	DELETE Alerts Key Rotation ID below
/Alerts/KeyRotation/{id}	GET	Retrieves details for the SSH key rotation alert for the specified ID.	GET Alerts Key Rotation ID on the next page
/Alerts/KeyRotation/Schedule	GET	Retrieves details of the schedule for delivery of SSH key rotation alerts.	GET Alerts Key Rotation Schedule on page 136
/Alerts/KeyRotation/Schedule	PUT	Updates the schedule for delivery of SSH key rotation alerts.	PUT Alerts Key Rotation Schedule on page 138
/Alerts/KeyRotation	GET	Retrieves details for all configured SSH key rotation alerts.	GET Alerts Key Rotation on page 140
/Alerts/KeyRotation	POST	Creates a new SSH key rotation alert.	POST Alerts Key Rotation on page 143
/Alerts/KeyRotation	PUT	Updates the SSH key rotation alert for a specified ID.	PUT Alerts Key Rotation on page 150
/Alerts/KeyRotation/Test	POST	Used to test specific SSH key rotation alerts.	POST Alerts Key Rotation Test on page 157
/Alerts/KeyRotation/TestAll	POST	Used to test all SSH key rotation alerts.	POST Alerts Key Rotation Test All on page 159

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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 83: 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 140) to retrieve a list of all the SSH key rotation alerts to determine the alert ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 84: 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 140) to retrieve a list of all the SSH key rotation alerts to determine the alert ID.

Table 85: 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>{server-logons}</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 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</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the file-name and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Web API Common Features on page 3).

Tip: The following permissions (see Security Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 86: GET Alerts Key Rotation Schedule Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Schedule</td><td>An array indicating the schedule for delivery of the SSH key rotation alerts. Possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature: WorkflowManagement: *Modify*

Table 87: PUT Alerts Key Rotation Schedule Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Schedule</td><td>Body</td><td>An array indicating the schedule for delivery of the SSH key rotation alerts. Possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

Table 88: PUT Alerts Key Rotation Schedule Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Schedule</td><td>An array indicating the schedule for delivery of the SSH key rotation alerts. Possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature: WorkflowManagement: *Read*

Table 89: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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.
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 90: 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 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</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the file-name and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 91: 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 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	Body	An integer indicating the number of days prior to the end of an SSH key's lifetime the alert should be triggered.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table>
<tr><th>ID</th><th>Event Handler Type</th></tr>
<tr><td>10</td><td>SSHKeyRotationLogger</td></tr>
<tr><td>11</td><td>SSHKeyRotationPowershell</td></tr>
</table></td></tr>
<tr><td>UseHandler</td><td>A Boolean indicating whether event handler use is enabled for the alert (true) or not (false).</td></tr>
</table>
For more information about event handlers, see Using Event Handlers in the Keyfactor Command Reference Guide.</td></tr>
<tr><td>EventHandlerParameters</td><td>Body</td><td>An array 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</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 PowerShell handler and is used to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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" }, { "Id": 29, "Key": "comment", "DefaultValue": "comment", "ParameterType": "Token" }, { "Id": 30, "Key": "Text", "DefaultValue": "Key Rotation Alert: 3 Days", "ParameterType": "Value" }, { "Id": 32, "Key": "ScriptName",</pre>

Name	In	Description
		"DefaultValue": "MyScript.ps1", "ParameterType": "Script" }]

Table 92: 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>{server-logons}</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 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
</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the file-name and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Warning: 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 93: 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 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	Body	An integer indicating the number of days prior to the end of an SSH key's lifetime the alert should be triggered.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RegisteredEventHandler</td><td>Body</td><td>An object containing the event handler configuration for the alert, if applicable. Possible values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>Id</td><td>An integer indicating the Keyfactor Command reference ID for the event handler.
<table><tr><th>ID</th><th>Event Handler Type</th></tr><tr><td>10</td><td>SSHKeyRotationLogger</td></tr><tr><td>11</td><td>SSHKeyRotationPowershell</td></tr></table></td></tr><tr><td>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 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</td></tr></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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" }, { "Id": 29, "Key": "comment", "DefaultValue": "comment", "ParameterType": "Token" }, { "Id": 30, "Key": "Text", "DefaultValue": "Key Rotation Alert: 3 Days", "ParameterType": "Value" }, {</pre>

Name	In	Description
		<pre>"Id": 32, "Key": "ScriptName", "DefaultValue": "MyScript.ps1", "ParameterType": "Script" }]</pre>

Table 94: 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>{server-logons}</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 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
</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>to reference the PowerShell script that should be run when the alert is triggered. It is referenced using the file-name and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. • 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*
WorkflowManagement: *Test*

Table 95: POST Alerts Key Rotation Test Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>keyRotationAlertTestRequest</td><td>Body</td><td>Required. An array containing information for the alert test. Alert test detail values are:
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>AlertId</td><td>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 140) to retrieve a list of all your key rotation alerts to determine the alert Id.</td></tr>
<tr><td>EvaluationDate</td><td>Required. A string indicating the start date/time for the test, in UTC.
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.</td></tr>
<tr><td>PreviousEvaluationDate</td><td>Required. A string indicating the end date/time for the test, in UTC.</td></tr>
<tr><td>SendAlerts</td><td>A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is false.</td></tr>
</table>
For example:
<pre>
{
   "EvaluationDate": "2022-08-31T20:51:33.528Z",
   "PreviousEvaluationDate": "2022-08-
31T20:51:33.528Z",
   "SendAlerts": true
}
</pre></td></tr>
</table>

Table 96: POST Alerts Key Rotation Test 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*
WorkflowManagement: *Test*

Table 97: POST Alerts Key Rotation Test All Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>keyRotationAlertTestRequest</td><td>Body</td><td>Required. An array containing information for the alert test. Alert test detail values are:

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>EvaluationDate</td><td>Required. A string indicating the start date/time for the test, in UTC.

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.</td></tr>
<tr><td>PreviousEvaluationDate</td><td>Required. A string indicating the end date/time for the test, in UTC.</td></tr>
<tr><td>SendAlerts</td><td>A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is false.</td></tr>
</table>

For example:

<pre>{
   "EvaluationDate": "2022-08-31T20:51:33.528Z",
   "PreviousEvaluationDate": "2022-08-
31T20:51:33.528Z",
   "SendAlerts": true
}</pre></td></tr>
</table>

Table 98: POST Alerts Key Rotation Test All Response Data

Parameter	Description
KeyRotationAlerts	An object containing alert details resulting from the test. Expiration alert details are: **Name** / **Description** Subject — A string indicating the subject for the email message, including any replaced substitutable special text. Message — A string indicating the email message, including any replaced substitutable special text. See *Table: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.4.5 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 1259).
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 99: Alerts Pending

Endpoint	Method	Description	Link
/Alerts/Pending/{id}	DELETE	Deletes a pending certificate request alert for the specified ID.	DELETE Alerts Pending ID on the next page

Endpoint	Method	Description	Link
/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 184
/Alerts/Pending/Schedule	GET	Retrieves details of the schedule for delivery of pending certificate request alerts.	GET Alerts Pending Schedule on page 167
/Alerts/Pending/Schedule	PUT	Updates the schedule for delivery of pending certificate request alerts.	PUT Alerts Pending Schedule on page 169
/Alerts/Pending	GET	Retrieves details for all configured pending certificate request alerts.	GET Alerts Pending on page 172
/Alerts/Pending	POST	Creates a new pending certificate request alert.	POST Alerts Pending on page 176
/Alerts/Pending/Test	POST	Tests all alerts	POST Alerts Pending TestAll on page 194
/Alerts/Pending/Test/{id}	POST	Tests specific alerts	POST Alerts Pending Test on page 192

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 Overview) are required to use this feature: WorkflowManagement: *Modify*

Table 100: 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 172) to retrieve a list of all the pending request alerts to determine the alert ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 101: 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 172) to retrieve a list of all the pending request alerts to determine the alert ID.

Table 102: GET Alerts Pending {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the pending request alert. Run GET /Alerts/Pending to find the pending request alert ID.
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<t-able>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:Ap-pOwnerFirstName}</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:Ap-pOwnerEmailAddress}</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 Certi-ficate 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. Avail-able 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>Description</th></tr>
<tr><td></td><td>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>
<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></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Web API Common Features on page 3).

Tip: The following permissions (see Security Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 103: GET Alerts Pending Schedule Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Schedule</td><td>An array indicating the schedule for delivery of the pending request alerts. Possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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. This method has no input parameters other than the standard headers (see Web API Common Features on page 3).

Tip: The following permissions (see Security Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 104: PUT Alerts Pending Schedule Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Schedule</td><td>Body</td><td>An array indicating the schedule for delivery of the pending request alerts. Possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

Table 105: PUT Alerts Pending Schedule Response Data

Name	Description
Schedule	An array 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. **Interval.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: **Daily.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. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }`

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 106: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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.
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 107: GET Alerts Pending Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the pending request alert. Run GET /Alerts/Pending to find the pending request alert ID.
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<t-able>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:Ap-pOwnerFirstName}</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:Ap-pOwnerEmailAddress}</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 Certi-ficate 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. Avail-able 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>Description</th></tr>
<tr><td></td><td>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>
<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></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 108: 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: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	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>• 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 GET Templates on page 1206) 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>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 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</td></tr></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td>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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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",</pre>

Name	In	Description
		"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" }]
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 109: POST Alerts Pending Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the pending request alert. Run GET /Alerts/Pending to find the pending request alert ID.
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<t-able>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:Ap-pOwnerFirstName}</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:Ap-pOwnerEmailAddress}</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 Certi-ficate 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. Avail-able 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>Description</th></tr>
<tr><td></td><td>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>
<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></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Warning: 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 110: PUT Alerts Pending Input Parameters

Name	In	Description
id	Path	An integer indicating the Keyfactor Command reference ID of the pending request alert. Run the
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: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	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}

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>The certificate requester, based on a lookup in Active Directory of the email address associated with the requester on the certificate.
• Your custom email-based metadata field, which would be specified similarly to {metadata:AppOwnerEmailAddress}.</td></tr>
<tr><td>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 GET Templates on page 1206) 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>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 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></table></td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server. - 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,</pre>

Name	In	Description
		"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" }]

Table 111: PUT Alerts Pending Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the pending request alert. Run GET /Alerts/Pending to find the pending request alert ID.
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<t-able>\n<tr><th>Certificate Details</th><th>Metadata</th></tr>\n<tr><td>CN: {rcn}</td><td>App Owner First Name: {metadata:Ap-pOwnerFirstName}</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:Ap-pOwnerEmailAddress}</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 Certi-ficate 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. Avail-able 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>Description</th></tr>
<tr><td></td><td>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>
<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></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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 filename and optional relative path to the PowerShell script, located in the extensions directory on the Keyfactor Command server.
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>
</table>

Name	Description
CommonName	A string indicating the common name of the certificate.
LogicalName	A string indicating the logical name of the certificate authority.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*
WorkflowManagement: *Test*

Table 112: POST Alerts Pending Test Input Parameters

<table>
<tr><th>Parameter</th><th>In</th><th>Description</th></tr>
<tr><td>req</td><td>Body</td><td>Required. An array containing information for the alert test. Alert test detail values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>AlertId</td><td>An integer indicating the Keyfactor Command reference ID for the pending alert.</td></tr><tr><td>SendAlerts</td><td>A Boolean indicating whether to send alert emails with the test (true), or not (false).</td></tr></table>For example:
<pre>{
   "AlertId": 1,
   "SendAlertEmails": false
}</pre></td></tr>
</table>

Table 113: 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</td></tr></table></td></tr>
</table>

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 object 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
WorkflowManagement: *Read*
WorkflowManagement: *Test*

Table 114: POST Alerts Pending Test All Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>req</td><td>Body</td><td>Required. An array containing information for the alert test. Alert test detail values are:
<table><tr><th>Value</th><th>Description</th></tr><tr><td>SendAlerts</td><td>A Boolean indicating whether to send alert emails with the test (true), or not (false).</td></tr></table>
For example:
<pre>{
   "SendAlertEmails": false
}</pre></td></tr>
</table>

Table 115: POST Alerts Pending Test All Response Data

<table>
<tr><th>Name</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 object 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>An integer indicating the number of pending alerts run by the test.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.5 Audit

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

Table 116: 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 201
/	GET	Returns a list of all audit log entries according to the provided filters and input parameters.	GET Audit on page 202
/Download	GET	Returns a comma separated list of audit log entries according to the provided filters and input parameters.	GET Audit Download on page 207
/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 211

2.2.5.1 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 Overview) are required to use this feature:
Auditing: *Read*

Table 117: 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 202) to retrieve a list of all the audit log entries to determine the audit log entry ID.

Table 118: 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	AuditingCertificateScheduledReplacement	Auditing Certificate Scheduled Replacement
2001	AuditingCertificateRequest	Certificate Request
2002	ApiApplication	API Application
2003	Template	Template
2004	CertificateQuery	Certificate Collection/Query
2005	ExpirationAlert	Expiration Alert
2005	ExpirationAlertDefinitionContextModel	Expiration Alert
2006	PendingAlert	Pending Alert
2006	PendingAlertDefinitionContextModel	Pending Alert
2007	ApplicationSetting	Application Setting
2008	IssuedAlert	Issued Alert
2008	IssuedAlertDefinitionContextModel	Issued Alert
2009	DeniedAlert	Denied Alert
2009	DeniedAlertDefinitionContextModel	Denied Alert

Name	Description

Value	Subcategory Name	Description
2010	ADIdentityModel	Security Identity
2011	SecurityRole	Security Role
2012	AuthorizationFailure	Authorization Failure
2013	CertificateSigningRequest	CSR
2014	ServerGroup	SSH Server Group
2015	Server	SSH Server
2016	DiscoveredKey	Rogue Key for Logon
2016	Key	SSH Key
2017	ServiceAccount	SSH Service Account
2018	Logon	SSH Logon
2019	SshUser	SSH User
2020	KeyRotationAlertDefinitionContextModel	SSH Key Rotation Alert
2021	CertificateStore	Certificate Store
2022	JobType	Orchestrator Job Type
2023	AgentSchedule	Orchestrator Job
2024	BulkAgentSchedule	Bulk Orchestrator Job
2025	CertificateStoreContainer	Store Container
2026	Agent	Orchestrator
2027	RevocationMonitoring	Monitoring
2028	License	License
2029	WorkflowDefinition	Workflow Definition
2030	WorkflowInstance	Workflow Instance
2031	WorkflowInstanceSignal	Workflow Instance Signal

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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:
<code>category -contains "Agent"</code></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><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></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>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).</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.5.2 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 Overview) are required to use this feature:
Auditing: *Read*

Table 119: 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 below) to retrieve a list of all the audit log entries to determine the audit log entry ID.

Table 120: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.5.3 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 Overview) are required to use this feature:
Auditing: *Read*

Table 121: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Audit Log Search Feature*. The query fields supported for this endpoint are: • *Name* (EntityIdentifier) • *Category* (EntityType) (see Table 122: GET Audit Response Data for codes) • *ImmutableIdentifier* • *Level* (see Table 122: GET Audit Response Data for codes) • *Operation* (see Table 122: 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.
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 122: 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	AuditingCertificateScheduledReplacement	Auditing Certificate Scheduled Replacement
2001	AuditingCertificateRequest	Certificate Request
2002	ApiApplication	API Application
2003	Template	Template
2004	CertificateQuery	Certificate Collection/Query
2005	ExpirationAlert	Expiration Alert
2005	ExpirationAlertDefinitionContextModel	Expiration Alert
2006	PendingAlert	Pending Alert
2006	PendingAlertDefinitionContextModel	Pending Alert
2007	ApplicationSetting	Application Setting
2008	IssuedAlert	Issued Alert
2008	IssuedAlertDefinitionContextModel	Issued Alert
2009	DeniedAlert	Denied Alert
2009	DeniedAlertDefinitionContextModel	Denied Alert

Name	Description

Value	Subcategory Name	Description
2010	ADIdentityModel	Security Identity
2011	SecurityRole	Security Role
2012	AuthorizationFailure	Authorization Failure
2013	CertificateSigningRequest	CSR
2014	ServerGroup	SSH Server Group
2015	Server	SSH Server
2016	DiscoveredKey	Rogue Key for Logon
2016	Key	SSH Key
2017	ServiceAccount	SSH Service Account
2018	Logon	SSH Logon
2019	SshUser	SSH User
2020	KeyRotationAlertDefinitionContextModel	SSH Key Rotation Alert
2021	CertificateStore	Certificate Store
2022	JobType	Orchestrator Job Type
2023	AgentSchedule	Orchestrator Job
2024	BulkAgentSchedule	Bulk Orchestrator Job
2025	CertificateStoreContainer	Store Container
2026	Agent	Orchestrator
2027	RevocationMonitoring	Monitoring
2028	License	License
2029	WorkflowDefinition	Workflow Definition
2030	WorkflowInstance	Workflow Instance
2031	WorkflowInstanceSignal	Workflow Instance Signal

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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:
<code>category -contains "Agent"</code></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><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></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>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).</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.5.4 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 Overview) are required to use this feature:
Auditing: *Read*

Table 123: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Audit Log Search Feature*. The query fields supported for this endpoint are: • *Name* (EntityIdentifier) • *Category* (EntityType) (see Table 122: GET Audit Response Data for codes) • *ImmutableIdentifier* • *Level* (see Table 122: GET Audit Response Data for codes) • *Operation* (see Table 122: 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.
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 124: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.5.5 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 Overview) are required to use this feature:
Auditing: *Read*

Table 125: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Audit Log Search Feature*. The query fields supported for this endpoint are: • *Name* (EntityIdentifier) • *Category* (EntityType) (see Table 122: GET Audit Response Data for codes) • *ImmutableIdentifier* • *Level* (see Table 122: GET Audit Response Data for codes) • *Operation* (see Table 122: 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.
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 126: GET Audit Related Entities Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>The ID of the specified audit log entry.</td></tr>
<tr><td>TimeStamp</td><td>The timestamp (UTC) on the audit log entry indicating when the action performed occurred.</td></tr>
<tr><td>Message</td><td>XML data on the audit event.</td></tr>
<tr><td>Signature</td><td>The signature on the audit entry.</td></tr>
<tr><td>Category</td><td>An integer identifying the category of the audit entry. Possible values are:
<table>
<tr><th>Value</th><th>Subcategory Name</th><th>Description</th></tr>
<tr><td>2001</td><td>Certificate</td><td>Certificate</td></tr>
<tr><td>2001</td><td>AuditingCertificateScheduledReplacement</td><td>Auditing Certificate Scheduled Replacement</td></tr>
<tr><td>2001</td><td>AuditingCertificateRequest</td><td>Certificate Request</td></tr>
<tr><td>2002</td><td>ApiApplication</td><td>API Application</td></tr>
<tr><td>2003</td><td>Template</td><td>Template</td></tr>
<tr><td>2004</td><td>CertificateQuery</td><td>Certificate Collection/Query</td></tr>
<tr><td>2005</td><td>ExpirationAlert</td><td>Expiration Alert</td></tr>
<tr><td>2005</td><td>ExpirationAlertDefinitionContextModel</td><td>Expiration Alert</td></tr>
<tr><td>2006</td><td>PendingAlert</td><td>Pending Alert</td></tr>
<tr><td>2006</td><td>PendingAlertDefinitionContextModel</td><td>Pending Alert</td></tr>
<tr><td>2007</td><td>ApplicationSetting</td><td>Application Setting</td></tr>
<tr><td>2008</td><td>IssuedAlert</td><td>Issued Alert</td></tr>
<tr><td>2008</td><td>IssuedAlertDefinitionContextModel</td><td>Issued Alert</td></tr>
<tr><td>2009</td><td>DeniedAlert</td><td>Denied Alert</td></tr>
<tr><td>2009</td><td>DeniedAlertDefinitionContextModel</td><td>Denied Alert</td></tr>
</table>
</td></tr>
</table>

Name	Description

Value	Subcategory Name	Description
2010	ADIdentityModel	Security Identity
2011	SecurityRole	Security Role
2012	AuthorizationFailure	Authorization Failure
2013	CertificateSigningRequest	CSR
2014	ServerGroup	SSH Server Group
2015	Server	SSH Server
2016	DiscoveredKey	Rogue Key for Logon
2016	Key	SSH Key
2017	ServiceAccount	SSH Service Account
2018	Logon	SSH Logon
2019	SshUser	SSH User
2020	KeyRotationAlertDefinitionContextModel	SSH Key Rotation Alert
2021	CertificateStore	Certificate Store
2022	JobType	Orchestrator Job Type
2023	AgentSchedule	Orchestrator Job
2024	BulkAgentSchedule	Bulk Orchestrator Job
2025	CertificateStoreContainer	Store Container
2026	Agent	Orchestrator
2027	RevocationMonitoring	Monitoring
2028	License	License
2029	WorkflowDefinition	Workflow Definition
2030	WorkflowInstance	Workflow Instance
2031	WorkflowInstanceSignal	Workflow Instance Signal

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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:
<code>category -contains "Agent"</code></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><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></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>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).</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6 Certificates

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

Table 127: 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 the next page
/{id}/Validate	GET	Validates that a certificate chain can be built for the specified certificate.	GET Certificates ID Validate on page 219
/Locations/{id}	GET	Returns details about the certificates stores in which the certificate is located.	GET Certificates Locations ID on page 224
/IdentityAudit/{id}	GET	Returns audit identity permissions for certificate.	GET Certificates Identity Audit ID on page 227
/CSV	GET	Returns content, in a CSV format, of certificates from Keyfactor Command that match the query criteria provided in the body.	GET Certificates CSV on page 229
/{id}	DELETE	Deletes a certificate from the Keyfactor Command database by its ID.	DELETE Certificates ID on page 231
/{id}	GET	Returns certificate details for a specified certificate.	GET Certificates ID on page 232
/Metadata/Compare	GET	Compares the metadata value provided with the metadata value associated with the specified certificate.	GET Certificates Metadata Compare on page 243
/{id}/History	GET	Returns the certificate operations history for a specified certificate.	GET Certificates ID History on page 244
/	DELETE	Deletes multiple certificates from the Keyfactor Command database, as specified by the IDs in the request body.	DELETE Certificates on page 246
/	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 247
/Metadata	PUT	Updates the metadata for a specified certificate.	PUT Certificates Metadata on page 261
/Metadata/All	PUT	Updates the metadata for an array of certificate IDs.	PUT Certificates Metadata All on

Endpoint	Method	Description	Link
			page 262
/Import	POST	Imports a certificate into Keyfactor Command.	POST Certificates Import on page 265
/Revoke	POST	Revokes a certificate.	POST Certificates Revoke on page 268
/Analyze	POST	Reads a base-64 encoded PEM certificates and returns it in human-readable form.	POST Certificates Analyze on page 270
/Recover	POST	Returns a recovered certificate as a PFX.	POST Certificates Recover on page 271
/Download	POST	Downloads a certificate.	POST Certificates Download on page 273
/RevokeAll	POST	Revokes all the certificates in the provided query.	POST Certificates Revoke All on page 275
/Query	DELETE	Deletes multiple certificates from the Keyfactor Command database based on search query.	DELETE Certificates Query on page 277
/PrivateKey	DELETE	Deletes the stored private keys of multiple certificates within the Keyfactor Command database.	DELETE Certificates Private Key on page 278
/PrivateKey/{id}	DELETE	Deletes the stored private key(s) of a certificate within the Keyfactor Command database.	DELETE Certificates Private Key ID on page 278

2.2.6.1 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 Overview) are required to use this feature:
Certificates: *Read*
SecuritySettings: *Read*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 128: 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 129: GET Certificates {id} Security Response Data

Name	Description
Roles	An array 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** — 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.2 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 Overview) are required to use this feature:
Certificates: *Read*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 130: 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 131: GET Certificates {id} Validate Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Valid</td><td>A Boolean that indicates whether all the validity tests are in a passing state (true) or not (false).</td></tr>
<tr><td>Results</td><td>An array containing the X509 certificate chain validity fields. The included validity fields are:
<table>
<tr><th>Name</th><th>Keyfactor Command Management Portal Equivalent</th><th>Description</th></tr>
<tr><td>NotTimeValid</td><td>Time Valid</td><td>A value of Pass indicates that the certificate time value is valid. A time can appear invalid (Fail) for a certificate that has expired.</td></tr>
<tr><td>NotTimeNested</td><td>n/a</td><td>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.</td></tr>
<tr><td>Revoked</td><td>Active</td><td>A value of Pass indicates that the X509 certificate chain is valid for the certificate and contains no revoked certificates or errors.</td></tr>
<tr><td>NotSignatureValid</td><td>Signature</td><td>A value of Fail indicates that the X509 certificate chain is invalid as a result of an invalid certificate signature.</td></tr>
</table></td></tr>
</table>

Name	Description

Name	Keyfactor Command Management Portal Equivalent	Description
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	Keyfactor Command Management 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

Name	Description

Name	Keyfactor Command Management Portal Equivalent	Description
		built up to the root certificate.
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

Name	Description
	<table><tr><th>Name</th><th>Keyfactor Command Management Portal Equivalent</th><th>Description</th></tr><tr><td></td><td></td><td>design in the certificate or that if a group policy has specified that all certificates must have a certificate policy, the certificate policy exists in the certificate.</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.3 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 Overview) are required to use this feature:
Certificates: *Read*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 132: 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 133: GET Certificates Locations {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Details</td><td>An array containing the certificate stores in which the certificate is found. Certificate store details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreType</td><td>A string indicating the type of certificate store (e.g. Java Keystore).</td></tr>
<tr><td>StoreTypeId</td><td>An integer indicating the Keyfactor Command referenced ID for the type of certificate store.
Use the GET CertificateStoreTypes method (see <a>GET Certificate Store Types on page 526) to retrieve a list of all the certificate store types to see a complete list of types.</td></tr>
<tr><td>StoreCount</td><td>An integer indicating the number of stores of the type referenced by StoreType in which the certificate is found.</td></tr>
<tr><td>Locations</td><td>An array containing details about the specific certificate stores in which the certificate is found. The following details are included about each store:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreId</td><td>A GUID that identifies the certificate store in which the certificate is located.</td></tr>
<tr><td>StoreTypeId</td><td>An integer indicating the Keyfactor Command reference ID for the type of certificate store.</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. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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 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.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Name	Description
	Name / **Description** **Name** / **Description** /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. Alias / A string containing the alias of the certificate in the certificate store. The format for this path will vary depending on the certificate store type. For example, for a Java keystore, this will be a user-provided string, but for an IIS Personal store, this will be the thumbprint of the certificate. See *PFX Enrollment* in the *Keyfactor Command Reference Guide* for more information.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.4 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 Overview) are required to use this feature:
Certificates: *Read*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 134: 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.

Table 135: GET Certificates {id} History Response Data

Name	Description
Id	An integer containing the Keyfactor ID of the user/group.
AccountName	A string containing the name of the Keyfactor user/group.
IdentityType	A string that specifies if the account is a user or a group.
SID	A string containing the SID of the user/group
Permissions	An array of the permissions for the certificate.

Parameter	Description
Name	A string containing the name of the permission (for example: Read, EditMetadata, Import, Recover, etc...)
GrantedBy	A string containing the list of roles or collections that grant the given permission to the user-/group.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.5 GET Certificates CSV

The GET /Certificates/CSV method is used to create content, in a CSV format, of certificates from Keyfactor Command that match the query criteria provided in the body. The content will display in the response body and can be copied from there for use in a file, as needed. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: Global *Read* Users can query with and without collection ID.
Certificate Collection: Users without Global *Read* permissions MUST provide a collection ID for which they have *Read* permissions.[1]

Table 136: GET Certificates CSV Input Parameters

Name	In	Description
Sortname	Query	A string of the fieldname by which the results should be sorted.
SortOrder	Query	A Boolean to indicate the field sort direction [0=ascending, 1=descending] to apply to the SortName.
query	Query	A string query, to limit the requested set of certificates, in the form: `"CN -contains \"mycertificate.keyexample.com\""` See Certificate Search Page section of the *Keyfactor Command Reference Guide* for querying guidelines to build your body query.
CollectionId	Query	An integer of the certificate collection ID (used to determine user access to the certificates).

[1]The certificate collection ID is used 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 137: GET Certificates CSV Response Body

Description
A comma-separated sting of certificate data as per the criteria supplied. For example: "Issued DN,Import Date,Effective Date,Expiration Date,Issued CN,Certificate Authority Name,Template Display Name,Principal,Requester,Key Type,Key Size,Certificate State,Thumbprint\r\n\"CN=Root CA,DC=keyexample,DC=com\",\"2022-11-17T19:52:00.8900000Z\",\"2020-04-14T23:39:28.0000000Z\",\"2035-04-14T23:49:28.0000000Z\",\"Root CA\",\"CN=Root CA,DC=keyexample,DC=com\",\"\",\"\",\"KEYEXAMPLE\\SQL241$\",\"RSA\",\"2048\",\"CertificateAuthority\",\"6DEA35B694F008B7DD69AE88B349B2FFD8AFF503\",\r\n\"CN=dc240.keyexample.com\",\"2022-11-17T19:52:01.2230000Z\",\"2020-04-15T03:58:35.0000000Z\",\"2021-04-15T03:58:35.0000000Z\",\"dc240.keyexample.com\",\"CN=Root CA,DC=keyexample,DC=com\",\"DomainController\",\"\",\"KEYEXAMPLE\\DC240$\",\"RSA\",\"2048\",\"Active\",\"2F2407C5329C646AC75834EAACE65DBF70F8FB1B\",\r\n\"CN=sql241.keyexample.com\",\"2022-11-17T19:52:01.2530000Z\",\"2020-04-15T14:50:52.0000000Z\",\"2022-04-15T14:50:52.0000000Z\",\"sql241.keyexample.com\",\"CN=Root CA,DC=keyexample,DC=com\",\"EnterpriseWebServer\",\"\",\"KEYEXAMPLE\\SQL241$\",\"RSA\",\"2048\",\"Active\",\"FEB56E1F076CFD2AEFE6969885F1CD59CBC839DB\",\r\n\"CN=red-apple.keyexample.com\",\"2022-11-17T19:52:01.3000000Z\",\"2020-04-15T16:57:06.0000000Z\",\"2020-05-27T16:57:06.0000000Z\",\"red-apple.keyexample.com\",\"CN=Root CA,DC=keyexample,DC=com\",\"EnterpriseWebServer-ShortLifetime\",\"\",\"KEYEXAMPLE\\sarahd\",\"RSA\",\"2048\",\"Active\",\"D60EC19B68589885DC8BD60BAD7EC7E81BAC256B\",\r\n\"CN=keyfactor243.keyexample.com\",\"2022-11-17T19:52:01.3330000Z\",\"2020-04-15T17:42:51.0000000Z\",\"2022-04-15T17:42:51.0000000Z\",\"keyfactor243.keyexample.com\",\"CN=Root CA,DC=keyexample,DC=com\",\"EnterpriseWebServer\",\"\",\"KEYEXAMPLE\\SRVR243$\",\"RSA\",\"2048\",\"Active\",\"BFA679661A7F40A507F9C518BE2A8CFF6D364AA0\",\r\n\"CN=walrus-apple.keyexample.com,L=Independence,ST=OH,C=US\",\"2022-11-17T19:52:01.3630000Z\",\"2020-09-14T22:07:52.0000000Z\",\"2020-10-26T22:07:52.0000000Z\",\"walrus-apple.keyexample.com\",\"CN=Root CA,DC=keyexample,DC=com\",\"EnterpriseWebServer-ShortLifetime\",\"\",\"BUFFY\\sarahd\",\"RSA\",\"2048\",\"Active\",\"D48DAE96497B4995D27B10EBC4EA571DAE80E6C8\",\r\n\"CN=walrus-apple.keyexample.com,L=Independence,ST=OH,C=US\",\"2022-11-17T19:52:01.3900000Z\",\"2020-09-14T22:23:51.0000000Z\",\"2020-10-26T22:23:51.0000000Z\",\"walrus-apple.keyexample.com\",\"CN=Root CA,DC=keyexample,DC=com\",\"EnterpriseWebServer-ShortLifetime\",\"\",\"BUFFY\\sarahd\",\"RSA\",\"2048\",\"Active\",\"D3E5CC1A4DBEFF0CE14F2ADAF9B811B9B9306A1C\",\r\n\"CN=appsrvr17.keyexample.com,OU=IT,O=Key Example\\, Inc,L=Independence,ST=OH,C=US\",\"2022-11-17T19:52:01.4430000Z\",\"2020-09-18T16:57:37.0000000Z\",\"2022-09-18T16:57:37.0000000Z\",\..."

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.6 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 Overview) are required to use this feature:
Certificates: *Delete*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, 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 138: 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 247) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.7 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 Overview) are required to use this feature:
Certificates: *Read*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 139: 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 247) 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 140: 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 *PrinicpalName*.
TemplateId	An integer indicating the Keyfactor Command reference ID of the template associated with the certificate.
CertState	An integer specifying the state of the certificate. The possible values are:

Value	Description
0	Unknown
1	Active
2	Revoked
3	Denied
4	Failed
5	Pending
6	Certificate Authority
7	Parent Certificate Authority

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></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)
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>
</table>

<table>
<tr><th>Name</th><th>Description</th></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 name of the template that was used when issuing the certificate.</td></tr>
<tr><td>ArchivedKey</td><td>A Boolean that indicates whether the certificate has a key archived in the issuing CA (true) or not (false).</td></tr>
<tr><td>HasPrivateKey</td><td>A Boolean that indicates whether the certificate has a private key stored in Keyfactor Command (true) or not (false)</td></tr>
<tr><td>PrincipalName</td><td>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).</td></tr>
<tr><td>CertRequestId</td><td>An integer containing the Keyfactor Command reference ID of the certificate request.</td></tr>
<tr><td>RequesterName</td><td>A string containing the name of the identity that requested the certificate.</td></tr>
<tr><td>ContentBytes</td><td>A string containing the certificate as bytes.</td></tr>
<tr><td>ExtendedKeyUsages</td><td>An array containing the extended key usages associated with the certificate. Extended Key data includes:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer containing the Keyfactor Command reference ID of the extended key usage.</td></tr><tr><td>Oid</td><td>A string indicating the OID of the extended key usage.</td></tr><tr><td>DisplayName</td><td>A string indicating the name of the extended key usage.</td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectAltNameElements</td><td>An array containing the subject alternative name elements of the certificate. SAN 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 SAN Element.</td></tr>
<tr><td>Value</td><td>A string indicating the value of the SAN Element.</td></tr>
<tr><td>Type</td><td>An integer containing the type of SAN element. The possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Other Name</td></tr>
<tr><td>1</td><td>RFC 822 Name</td></tr>
<tr><td>2</td><td>DNS Name</td></tr>
<tr><td>3</td><td>X400 Address</td></tr>
<tr><td>4</td><td>Directory Name</td></tr>
<tr><td>5</td><td>Ediparty Name</td></tr>
<tr><td>6</td><td>Uniform Resource Identifier</td></tr>
<tr><td>7</td><td>IP Address</td></tr>
<tr><td>8</td><td>Registered Id</td></tr>
<tr><td>100</td><td>MS_NTPrincipalName</td></tr>
<tr><td>101</td><td>MS_NTDSReplication</td></tr>
<tr><td>999</td><td>Unknown</td></tr>
</table></td></tr>
<tr><td>ValueHash</td><td>A string indicating a hash of the SAN value.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CRLDistributionPoints</td><td>An array 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 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 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 certificate was discovered.</td></tr>
<tr><td>Port</td><td>An integer indicating the port on which the certificate was discovered.</td></tr>
</table></td></tr>
</table>

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Locations</td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreMachine</td><td>A string indicating the machine on which the certificate store is located.</td></tr>
<tr><td>StorePath</td><td>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.</td></tr>
<tr><td>StoreType</td><td>An integer indicating the type of certificate store the certificate is located in. 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>
<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 in the certificate store.</td></tr>
<tr><td>ChainLevel</td><td>An integer stating how many certificates are below this certificate in the certificate chain stored at the given location.</td></tr>
</table>
</td></tr>
</table>

Name	Description
Metadata	An array 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 array 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 agreement.
KeyCertSign	A Boolean that indicates whether the certificate is configured for certificate signing.
KeyEncipherment	A Boolean that indicates whether the certificate 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>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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.8 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 Overview) are required to use this feature:
Certificates: *Read*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 141: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.9 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 Overview) are required to use this feature:
Certificates: *Read*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 142: 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.
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.
query.returnLimit	Query	An integer that specifies how many results to return per page. The default is 50.
query.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*.
query.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 143: 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 Update`

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.10 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.

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Delete*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, 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 144: DELETE Certificates Input Parameters

Name	In	Description
ids	Body	**Required**. Array of Keyfactor Command certificate IDs for certificates that should be deleted in the form: `[123,789,567]`
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.11 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 Overview) are required to use this feature:
Certificates: Global *Read*, or Collection ID *Read*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 145: 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*.

Name	In	Description
queryString	Query	• *ArchivedKey* • *CertId* • *CA* • *CertState* • *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* • *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*

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 146: 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 *PrinicpalName*.
TemplateId	An integer indicating the Keyfactor Command reference ID of the template associated with the certificate.
CertState	An integer specifying the state of the certificate. The possible values are:

Value	Description
0	Unknown
1	Active
2	Revoked
3	Denied
4	Failed
5	Pending
6	Certificate Authority
7	Parent Certificate Authority

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></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)
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>
</table>

<table>
<tr><th>Name</th><th>Description</th></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 name of the template that was used when issuing the certificate.</td></tr>
<tr><td>ArchivedKey</td><td>A Boolean that indicates whether the certificate has a key archived in the issuing CA (true) or not (false).</td></tr>
<tr><td>HasPrivateKey</td><td>A Boolean that indicates whether the certificate has a private key stored in Keyfactor Command (true) or not (false)</td></tr>
<tr><td>PrincipalName</td><td>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).</td></tr>
<tr><td>CertRequestId</td><td>An integer containing the Keyfactor Command reference ID of the certificate request.</td></tr>
<tr><td>RequesterName</td><td>A string containing the name of the identity that requested the certificate.</td></tr>
<tr><td>ContentBytes</td><td>A string containing the certificate as bytes.</td></tr>
<tr><td>ExtendedKeyUsages</td><td>An array containing the extended key usages associated with the certificate. Extended Key data includes:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>An integer containing the Keyfactor Command reference ID of the extended key usage.</td></tr><tr><td>Oid</td><td>A string indicating the OID of the extended key usage.</td></tr><tr><td>DisplayName</td><td>A string indicating the name of the extended key usage.</td></tr></table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectAltNameElements</td><td>An array containing the subject alternative name elements of the certificate. SAN 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 SAN Element.</td></tr>
<tr><td>Value</td><td>A string indicating the value of the SAN Element.</td></tr>
<tr><td>Type</td><td>An integer containing the type of SAN element. The possible values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Other Name</td></tr>
<tr><td>1</td><td>RFC 822 Name</td></tr>
<tr><td>2</td><td>DNS Name</td></tr>
<tr><td>3</td><td>X400 Address</td></tr>
<tr><td>4</td><td>Directory Name</td></tr>
<tr><td>5</td><td>Ediparty Name</td></tr>
<tr><td>6</td><td>Uniform Resource Identifier</td></tr>
<tr><td>7</td><td>IP Address</td></tr>
<tr><td>8</td><td>Registered Id</td></tr>
<tr><td>100</td><td>MS_NTPrincipalName</td></tr>
<tr><td>101</td><td>MS_NTDSReplication</td></tr>
<tr><td>999</td><td>Unknown</td></tr>
</table></td></tr>
<tr><td>ValueHash</td><td>A string indicating a hash of the SAN value.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>CRLDistributionPoints</td><td>An array 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 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 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 certificate was discovered.</td></tr><tr><td>Port</td><td>An integer indicating the port on which the certificate was discovered.</td></tr></table></td></tr>
</table>

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Locations</td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreMachine</td><td>A string indicating the machine on which the certificate store is located.</td></tr>
<tr><td>StorePath</td><td>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.</td></tr>
<tr><td>StoreType</td><td>An integer indicating the type of certificate store the certificate is located in. 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>
<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 in the certificate store.</td></tr>
<tr><td>ChainLevel</td><td>An integer stating how many certificates are below this certificate in the certificate chain stored at the given location.</td></tr>
</table>
</td></tr>
</table>

Name	Description
Metadata	An array 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 array 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 agreement.
KeyCertSign	A Boolean that indicates whether the certificate is configured for certificate signing.
KeyEncipherment	A Boolean that indicates whether the certificate 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>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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.12 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 Overview) are required to use this feature:
Certificates: *EditMetadata*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 147: PUT Certificates Metadata Input Parameters

Name	In	Description
Id	Body	**Required**. An integer specifying the Keyfactor Command reference ID for the certificate

Name	In	Description
		to update.
Metadata	Body	**Required**. An array 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", // This is String field.` `"SiteCode":23, // This is` `"BusinessCritical":true, // This is` `"Notes":"Here are some notes about this certificate.", // This is a BigText field.` `"BusinessUnit":"E-Business", // This is a Multiple Choi with a pre-defined value.` `"TicketResolutionDate":"2021-07-23" // This is a 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.13 PUT Certificates Metadata All

The PUT /Certificates/Metadata/All method is used to update one or more metadata values for a specified set of active certificates. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *EditMetadata*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 148: 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 the *Keyfactor Command Reference Guide* Certificate Search Page section. A value for one of *CertificateIds*, *Query*, or *CollectionId* is **required**. The query fields supported for this endpoint are: • *ArchivedKey* • *CertId* • *CA* • *CertState* • *CertStoreFQDN* (alias: *JavaKeystoreFQDN*) • *CertStorePath* (alias: *JavaKeystorePath*) • *CN* (alias: *IssuedCN*) • *DN* (alias: *IssuedDN*) • *ExpirationDate* (alias: *NotAfter*) • *EKU* • *EKUName* • *HasPrivateKey* • *ImportDate* • *IssuedDate* (aliases: *NotBefore* and *EffectiveDate*) • *IssuerDN* • *KeySize* (alias: *KeySizeInBits*) • *KeyType* • *KeyUsage* • *OU* • *NetBIOSPrincipal* (alias: *PrincipalName*) • *PublicKey* • *NetBIOSRequester* (alias: *RequesterName*) • *RevocationDate* (alias: *RevocationEffDate*) • *Revoker* • *RFC2818Compliant* • *SelfSigned* • *SerialNumber* • *SigningAlgorithm* • *SSLDNSName* • *SSLIPAddress* (alias: *SslHostName*) • *SSLNetworkName* • *SSLPort* • *SAN* • *TemplateDisplayName* (alias: *TemplateName*) • *TemplateShortName* • *Thumbprint* The following fields have been deprecated and will be ignored if included in a request: • *CARequestID* • *CertRequestId*

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

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:
<code>CertState -ne \"2\"</code>
To exclude expired certificates from the update, include a query of:
<code>ExpirationDate -ge \"%TODAY%\"</code>
</td></tr>
<tr><td>Certificate-Ids</td><td>Body</td><td>Required*. An array of Keyfactor Command certificate IDs to update. A value for one of CertificateIds, Query, or CollectionId is required.</td></tr>
<tr><td>Metadata</td><td>Body</td><td>Required. An array 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. The value that should be set for the metadata field.</td></tr>
<tr><td>MetadataName</td><td>Required. 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
</pre>
</td></tr>
</table>

Name	In	Description
		<pre> }, { "MetadataName": "BusinessCritical", // This is a Boolean field. "Value": true, "OverwriteExisting": true }, { "MetadataName": "Notes", // This is a BigText field. "Value": "Here are some notes about this certificate.", "OverwriteExisting": true }, { "MetadataName": "BusinessUnit", // This is a Multiple Choice field. "Value": "E-Business", // This is a value pre-defined for the field. "OverwriteExisting": true }, { "MetadataName": "TicketResolutionDate", // This is a Date field in yyyy-mm-dd format. "Value": "2021-07-23", "OverwriteExisting": true }]</pre>
Collec-tionId	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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.14 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 Overview) are required to use this feature:
Certificates: *Import*

Table 149: POST Certificates Import Input Parameters

Name	In	Description
Certificate	Body	**Required**. The base-64 encoded contents of the certificate that is to be imported into Keyfactor Command.
Password	Body	**Required***. The password used to decrypt the imported PFX. This field is **required** if a PFX certificate is provided in the *Certificate* field.
Metadata	Body	A list of 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	A list of the certificate store GUIDs that the imported certificate will be installed into.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>StoreTypes</td><td>Body</td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StoreTypeId</td><td>The ID of the store type being used. There must be one for each type of store ID used. 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>
<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>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>Overwrite</td><td>A Boolean that sets whether a certificate in the store with the Alias provided should be overwritten with the certificate being imported (true) or not (false).
Use the GET Certificates/Locations/{id} method (see GET Certificates Locations ID on page 224) 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>
</table>
</td></tr>
</table>

Name	In	Description
Schedule	Body	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. 2021-05-19T16:23:01Z).

Table 150: POST Certificates Import Response Data

Name	Description
ImportStatus	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	Which key store items failed with some information. Included parameters are: **Name** / **Description** KeystoreId — The ID of the store that failed. ClientMachine — The client machine of the store that failed. StorePath — The path to the location of the certificate store that failed. Alias — The alias for the certificate in the store that failed. Reason — The simple reason why it failed. Explanation — A more specific reason for the failure.
JobStatus	The state of all certificate store jobs.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.15 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

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Revoke*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, 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).

Table 151: 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 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>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. 2021-05-19T16:23:01Z). 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</td></tr>
</table>

Name	In	Description
		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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.16 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. This method returns HTTP 200 OK on a success with a list of the contents of the certificate.

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Read*
OR
Certificates: *Import*

Table 152: POST Certificates Analyze Input Parameters

Name	In	Description
Certificate	Body	**Required**. The base-64 encoded PEM string of the certificate, **not** including the header and footer (e.g. -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----).
Password	Body	The password used to encrypt the private key of the certificate for a base-64 encoded PEM containing the certificate's private key (-----BEGIN ENCRYPTED PRIVATE KEY-----).

Table 153: 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.
Thumbprint	A string containing the thumbprint of the certificate.
NotAfter	The date/time, in UTC, on which the certificate expires.
NotBefore	The date/time, in UTC, on which the certificate was issued by the certificate authority.
Metadata	An array containing the metadata fields populated for the certificate.
IsEndEntity	A boolean flag is marked true if the certificate is the end entity of the chain.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.17 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 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 273).

Tip: CA-level key recovery is supported for Microsoft CAs to allow recovery of private keys for certificates enrolled outside of Keyfactor Command. CA-level key archiving is not supported for enrollments done through Keyfactor Command. CA-level key recovery is not supported for EJBCA CAs. For enrollments done through Keyfactor Command for either Microsoft or EJBCA CAs, use Keyfactor Command private key retention (see Details Tab).

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Recover*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 154: POST Certificates Recover Input Parameters

Name	In	Description
Password	Body	**Required**. The password to set on the certificate.
CertID	Body	**Required*** . The Keyfactor Command reference ID of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
SerialNumber	Body	**Required*** . The serial number of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
IssuerDN	Body	**Required*** . 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*** . 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.
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.
x-certificateformat	Header	The desired output format for the certificate. Supported options are: • PEM • PFX

Table 155: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.18 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 271).

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Recover*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 156: POST Certificates Download Input Parameters

Name	In	Description
CertID	Body	**Required***. The Keyfactor Command reference ID of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
SerialNumber	Body	**Required***. The serial number of the certificate to retrieve. One of the following is **required**: • *CertID* • *Thumbprint* • *SerialNumber* and *IssuerDN*
IssuerDN	Body	**Required***. 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***. 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.
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.
x-certificateformat	Header	The desired output format for the certificate. Supported options are: • DER Not supported if IncludeChain is set to *true*. • PEM • P7B Only supported if IncludeChain is set to *true*

Table 157: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.19 POST Certificates Revoke All

The POST /Certificates/RevokeAll method is used to revoke all the certificates in the specified query or collection ID. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Revoke*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, 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).

Table 158: POST Certificates Revoke All Input Parameters

Name	In	Description
Query	Body	**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 the *Keyfactor Command Reference Guide* Certificate Search Page section. A value for either *Query* or *CollectionId* is **required**. If both *Query* and *CollectionId* are specified, certificates from both sources will be revoked.
Reason	Body	An integer containing the specific reason that the certificates are being revoked. Available values are:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>-1</td><td>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>
<tr><td>999</td><td>Unknown</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 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. 2021-05-19T16:23:01Z). The default is the current date and time.</td></tr>
<tr><td>IncludeRevoked</td><td>Body</td><td>A Boolean that indicates whether revoked certificates should be included in the revocation (true) or not (false). The default is false.</td></tr>
<tr><td>IncludeExpired</td><td>Body</td><td>A Boolean that indicates whether expired certificates should be included in the revocation (true) or not (false). The default is false.</td></tr>
<tr><td>CollectionId</td><td>Query</td><td>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.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.20 DELETE Certificates Query

The DELETE /Certificates/query method is used to delete a group of active 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 Overview) are required to use this feature:
Certificates: *Delete*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 159: 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. **Tip:** Revoked and expired certificates are excluded from the selection regardless of the query you enter.
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.21 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 Overview) are required to use this feature:
Certificates: *Delete*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 160: DELETE Certificates Private Key Input Parameters

Name	In	Description
ids	Body	**Required**. An array of Keyfactor Command reference IDs for certificates for which the associated private keys should be deleted in the form: `[123,789,567]`
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.6.22 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 Overview) are required to use this feature:
Certificates: *Delete*

Certificate permission can be granted at either the global or collection level. See note under CollectionId, below.

Table 161: 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 247) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.7 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 162: Certificate Authority Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the certificate authority record for the specified ID.	DELETE Certificate Authority ID on the next page
/{id}	GET	Returns details for the certificate authority identified by the specified ID.	GET Certificate Authority ID on the next page
/	GET	Returns a list of all certificate authorities.	GET Certificate Authority on page 292
/	POST	Creates a new certificate authority record.	POST Certificate Authority on page 305
/	PUT	Updates an existing certificate authority record.	PUT Certificate Authority on page 330

Endpoint	Method	Description	Link
/Test	POST	Validates that the certificate authority with the provided information can be reached.	POST Certificate Authority Test on page 356
/PublishCRL	POST	Publishes the Certificate Revocation List of the given certificate authority.	POST Certificate Authority PublishCRL on page 358

2.2.7.1 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 Overview) are required to use this feature:
PkiManagement: *Modify*

Note: You can't delete a CA from Keyfactor Command that has active records associated with it (e.g. certificates, certificate requests).

Table 163: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.7.2 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 Overview) are required to use this feature:
PkiManagement: *Read*

Table 164: 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 165: 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 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

Name	Description
	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 Windows 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 Windows Orchestrator or 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 Windows Orchestrator or 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 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.
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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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.</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.
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 <a>PUT Templates on page 1216).</td></tr>
<tr><td>Properties</td><td>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:
<code>{\"syncExternal\":true} OR {\"syncExternal\":false}</code></td></tr>
<tr><td>AllowedEnrollmentTypes</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>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></td></tr>
</table>

Name	Description
	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 1216). *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.
KeyRetentionDays	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	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	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

Name	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 Identities* 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	A string containing the password for the *ExplicitUser*.
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 1216).
AllowedRequesters	An array of Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example: `"AllowedRequesters": [` `  "Power Users",`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
   "Read Only"
]
</pre>
The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA.
This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA.
In addition to granting permissions at the CA level, you need to enable the UseAllowedRequesters option to grant permissions on a template-by-template basis (see PUT Templates on page 1216).
The values set here are only considered if UseAllowedRequesters is set to true.
</td></tr>
<tr><td>FullScan</td><td>
The schedule for the full synchronization of this certificate authority. 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>
</table>
</td></tr>
</table>

Name	Description
	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. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }` 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. 2021-05-19T16:23:01Z). 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": "2022-02-27T17:30:00Z" }` **Note:** Although the Swagger *Example Value* may show examples of various other

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
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 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>
The schedule for the incremental synchronization of this certificate authority. 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</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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Portal for this functionality—are valid for this endpoint.</td></tr>
<tr><td>ThresholdCheck</td><td>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:
<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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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>AuthCer-
tificatePassword</td><td>An array 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 PAM Providers on page 723 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 array 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 739) to retrieve a list of all the PAM providers to determine the ID.</td></tr></table>Due to its sensitive nature, this value is not returned in responses.</td></tr>
<tr><td>AuthCertificate</td><td>An array 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.
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:
<code>"IssuedDN": "CN=SuperAdmin,OU=IT,O=\"Key Example, Inc.\",L=Independence,ST=OH,C=US"</code></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>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>LastScan</td><td>A string indicating the date, in UTC, on which a synchronization was last performed for the CA.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.7.3 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 Overview) are required to use this feature:
PkiManagement: *Read*

Table 166: 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 167: 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 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

Name	Description
	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 Windows 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 Windows Orchestrator or 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 Windows Orchestrator or 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 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.
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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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.</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.
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 <a>PUT Templates on page 1216).</td></tr>
<tr><td>Properties</td><td>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:
<code>{\"syncExternal\":true} OR {\"syncExternal\":false}</code></td></tr>
<tr><td>AllowedEnrollmentTypes</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>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></td></tr>
</table>

Name	Description
	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 1216). *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.
KeyRetentionDays	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	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	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

Name	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 Identities* 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	A string containing the password for the *ExplicitUser*.
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 1216).
AllowedRequesters	An array of Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example: `"AllowedRequesters": [` `  "Power Users",`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
   "Read Only"
]
</pre>
The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA.
This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA.
In addition to granting permissions at the CA level, you need to enable the UseAllowedRequesters option to grant permissions on a template-by-template basis (see PUT Templates on page 1216).
The values set here are only considered if UseAllowedRequesters is set to true.
</td></tr>
<tr><td>FullScan</td><td>
The schedule for the full synchronization of this certificate authority. 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>
</table>
</td></tr>
</table>

Name	Description
	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. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }` 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. 2021-05-19T16:23:01Z). 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": "2022-02-27T17:30:00Z" }` **Note:** Although the Swagger *Example Value* may show examples of various other

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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>The schedule for the incremental synchronization of this certificate authority. 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</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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Portal for this functionality—are valid for this endpoint.</td></tr>
<tr><td>ThresholdCheck</td><td>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:
<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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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>AuthCer-
tificatePassword</td><td>An array 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 PAM Providers on page 723 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 array 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 739) to retrieve a list of all the PAM providers to determine the ID.</td></tr></table>Due to its sensitive nature, this value is not returned in responses.</td></tr>
<tr><td>AuthCertificate</td><td>An array 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.
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:
<code>"IssuedDN": "CN=SuperAdmin,OU=IT,O=\"Key Example, Inc.\",L=Independence,ST=OH,C=US"</code></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>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>LastScan</td><td>A string indicating the date, in UTC, on which a synchronization was last performed for the CA.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.7.4 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 Overview) are required to use this feature:
PkiManagement: *Modify*

Table 168: 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). 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.

Name	In	Description
Remote	Body	A Boolean that sets whether communications with the certificate authority are done via a Keyfactor Windows 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 Windows Orchestrator or 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 Windows Orchestrator or 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	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.
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.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>RFCEnforcement</td><td>Body</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.
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 1216).</td></tr>
<tr><td>Properties</td><td>Body</td><td>Required. 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:
<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></td></tr>
</table>

Name	In	Description
		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 1216). *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.
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

Name	In	Description
		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 Identities* 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	A string containing the password for the *ExplicitUser*.
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 1216).
AllowedRequesters	Body	An array of Keyfactor Command security roles that are allowed to enroll for certificates

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>via Keyfactor Command for this CA. For example:

<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 <a>PUT Templates on page 1216).
The values set here are only considered if UseAllowedRequesters is set to true.</td></tr>
<tr><td>FullScan</td><td>Body</td><td>The schedule for the full synchronization of this certificate authority. The following schedule types are supported:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>

For example, every hour:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. "Sunday").</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",</pre></td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><pre>"Wednesday", "Friday"], "Time": "2022-02-27T17:30:00Z" }</pre></td></tr></table> **Note:** Although the 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>"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

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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>The schedule for the incremental synchronization of this certificate authority. The following schedule types are supported:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Off</td><td>Turn off a previously configured schedule.</td></tr>
<tr><td>Interval</td><td>A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database.
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Minutes</td><td>An integer indicating the number of minutes between each interval.</td></tr>
</table>
For example, every hour:
<pre>"Interval": {
   "Minutes": 60
}</pre></td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</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": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the 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>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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>• 1—HTTPS</td></tr>
<tr><td>AuthCer-
tificatePassword</td><td>Body</td><td>An array 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 PAM Providers on page 723 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 array 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 739) 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:
<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 739 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</td></tr>
</table>

Name	In	Description
		in this example—is the Id value from GET PAM Providers on page 739 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 array 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*.
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.
LastScan	Body	A string indicating the date, in UTC, on which a synchronization was last performed for the CA.

Table 169: 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 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

Name	Description
	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 Windows 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 Windows Orchestrator or 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 Windows Orchestrator or 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 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.
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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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.</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.
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 <a>PUT Templates on page 1216).</td></tr>
<tr><td>Properties</td><td>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:
<code>{\"syncExternal\":true} OR {\"syncExternal\":false}</code></td></tr>
<tr><td>AllowedEnrollmentTypes</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>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></td></tr>
</table>

Name	Description
	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 1216). *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.
KeyRetentionDays	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	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	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

Name	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 Identities* 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	A string containing the password for the *ExplicitUser*.
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 1216).
AllowedRequesters	An array of Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example: `"AllowedRequesters": [` `  "Power Users",`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
   "Read Only"
]
</pre>
The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA.
This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA.
In addition to granting permissions at the CA level, you need to enable the UseAllowedRequesters option to grant permissions on a template-by-template basis (see PUT Templates on page 1216).
The values set here are only considered if UseAllowedRequesters is set to true.
</td></tr>
<tr><td>FullScan</td><td>
The schedule for the full synchronization of this certificate authority. 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>
</table>
</td></tr>
</table>

Name	Description
	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. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }` 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. 2021-05-19T16:23:01Z). 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": "2022-02-27T17:30:00Z" }` **Note:** Although the Swagger *Example Value* may show examples of various other

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 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>The schedule for the incremental synchronization of this certificate authority. 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</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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Portal for this functionality—are valid for this endpoint.</td></tr>
<tr><td>ThresholdCheck</td><td>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:
<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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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>AuthCer-
tificatePassword</td><td>An array 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 PAM Providers on page 723 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 array 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 739) to retrieve a list of all the PAM providers to determine the ID.</td></tr></table>Due to its sensitive nature, this value is not returned in responses.</td></tr>
<tr><td>AuthCertificate</td><td>An array 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.
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:
<code>"IssuedDN": "CN=SuperAdmin,OU=IT,O=\"Key Example, Inc.\",L=Independence,ST=OH,C=US"</code></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>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>LastScan</td><td>A string indicating the date, in UTC, on which a synchronization was last performed for the CA.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.7.5 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 Overview) are required to use this feature:
PkiManagement: *Modify*

Warning: 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 170: 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* 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

Name	In	Description
		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 Windows 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 Windows Orchestrator or 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 Windows Orchestrator or 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	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.
FailureMax	Body	An integer that sets the maximum number of certificate requests that can fail or be

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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.</td></tr>
<tr><td>RFCEnforcement</td><td>Body</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.
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 1216).</td></tr>
<tr><td>Properties</td><td>Body</td><td>Required. 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:
<pre>{\"syncExternal\":true} OR {\"syncExternal\":false}</pre></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></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>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 1216).
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>
<tr><td>ExplicitCredentials</td><td>Body</td><td>A Boolean that sets whether explicit credentials are enabled for this certificate authority (true) or not (false). Set this to true for CAs that 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.</td></tr>
<tr><td>SubscriberTerms</td><td>Body</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). 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</td></tr>
</table>

Name	In	Description
		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 Identities* 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	A string containing the password for the *ExplicitUser*.
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.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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 1216).</td></tr>
<tr><td>AllowedRequesters</td><td>Body</td><td>An array of Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example:<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 1216).
The values set here are only considered if UseAllowedRequesters is set to true.</td></tr>
<tr><td>FullScan</td><td>Body</td><td>The schedule for the full synchronization of this certificate authority. 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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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</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>time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the 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>"FullScan": {
   "Daily": {
      "Time": "2022-05-27T17:30:00Z"
   }
}</pre>
Or:
<pre>"FullScan": {
   "Weekly": {
      "Days": [
         "Monday",
         "Wednesday",
         "Friday"</pre>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<pre>
      ],
      "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.
</td></tr>
<tr><td>IncrementalScan</td><td>Body</td><td>
The schedule for the incremental synchronization of this certificate authority. 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": {
</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>
   "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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. "Sunday").</td></tr>
</table>

For example, every Monday, Wednesday and Friday at 5:30 pm:

<pre>
"Weekly": {
   "Days": [
      "Monday",
</pre></td></tr>
</table>
</td></tr>
</table>

<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>

```
        "Wednesday",
        "Friday"
    ],
    "Time": "2022-02-27T17:30:00Z"
}
```

</td></tr>
</table>
Note: Although the 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>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": {
   "Minutes": 60
}
```

</td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre>
</td></tr>
</table>

Note: Although the 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>AuthCer-tificatePassword</td><td>Body</td><td>An array 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 PAM Providers on page 723 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 array indicating the parameters to supply for PAM authen-</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></td><td>tication. 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 739) 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:
<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 739 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 739 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.
</td></tr>
<tr><td>AuthCertificate</td><td>Body</td><td>An array 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.</td></tr>
</table>

Name	In	Description
EnforceUniqueDN	Body	A Boolean that sets whether the unique DN requirement is enforced on the CA (*true*) or not (*false*). Checking this will cause Keyfactor Command, upon enrollment, to search the EJBCA CA for end entities with DNs that match the DN in the certificate request. If a matching DN is found, the process will update the existing end entity in EJBCA with the new certificate request information rather than creating a new end entity. If you enable this option in Keyfactor Command, it must also be enabled on the matching EJBCA CA. A mismatch in these settings can result in enrollment failures. This setting applies to HTTPS CAs only.
LastScan	Body	A string indicating the date, in UTC, on which a synchronization was last performed for the CA.

Table 171: 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 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

Name	Description
	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 Windows 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 Windows Orchestrator or 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 Windows Orchestrator or 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 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.
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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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.</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.
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 <a>PUT Templates on page 1216).</td></tr>
<tr><td>Properties</td><td>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:
<code>{\"syncExternal\":true} OR {\"syncExternal\":false}</code></td></tr>
<tr><td>AllowedEnrollmentTypes</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>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></td></tr>
</table>

Name	Description
	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 1216). *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.
KeyRetentionDays	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	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	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

Name	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 Identities* 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	A string containing the password for the *ExplicitUser*.
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 1216).
AllowedRequesters	An array of Keyfactor Command security roles that are allowed to enroll for certificates via Keyfactor Command for this CA. For example: `"AllowedRequesters": [` `  "Power Users",`

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
   "Read Only"
]
</pre>
The allowed requesters option is used to select Keyfactor Command security roles that a user must belong to in order to successfully enroll for certificates in Keyfactor Command via this CA.
This is used for EJBCA CAs and Microsoft CAs where integrated authentication is not supported. Integrated authentication is generally supported for Microsoft CAs, Keyfactor CA gateways, or Keyfactor CA management gateways on servers joined to the local Active Directory forest in which Keyfactor Command is installed and any Active Directory forests in a two-way trust with this forest. Since Keyfactor Command cannot make use of the access control model of the CA itself to determine which users can enroll for certificates at either a template or CA level without using integrated authentication, this setting replaces that functionality. This setting is similar to setting request certificates for the selected security roles at the CA level on a Microsoft CA.
In addition to granting permissions at the CA level, you need to enable the UseAllowedRequesters option to grant permissions on a template-by-template basis (see PUT Templates on page 1216).
The values set here are only considered if UseAllowedRequesters is set to true.
</td></tr>
<tr><td>FullScan</td><td>
The schedule for the full synchronization of this certificate authority. 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>
</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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
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 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>
The schedule for the incremental synchronization of this certificate authority. 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</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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>

Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Portal for this functionality—are valid for this endpoint.</td></tr>
<tr><td>ThresholdCheck</td><td>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:
<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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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>AuthCer-
tificatePassword</td><td>An array 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 PAM Providers on page 723 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 array 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 739) to retrieve a list of all the PAM providers to determine the ID.</td></tr></table>Due to its sensitive nature, this value is not returned in responses.</td></tr>
<tr><td>AuthCertificate</td><td>An array 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.
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:
<code>"IssuedDN": "CN=SuperAdmin,OU=IT,O=\"Key Example, Inc.\",L=Independence,ST=OH,C=US"</code></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>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>LastScan</td><td>A string indicating the date, in UTC, on which a synchronization was last performed for the CA.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.7.6 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 Overview) are required to use this feature:
PkiManagement: *Read*

Table 172: POST Certificate Authority Test 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).
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.
AuthCertificatePassword	Body	**Required***. An array indicating the password for the PKCS#12 client certificate to use to authenticate to the EJBCA CA. The password is provided in the following format: `{ "SecretValue": "MySuperSecretPassword" }` This parameter is required for EJBCA CAs.
AuthCertificate	Body	**Required***. An array containing the base-64 encoded PKCS#12 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 certificate is provided in the following format: `{ "SecretValue": "MIACAQMwgAY ... CAwGQAAAA" }` This parameter is required for EJBCA CAs.
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.

Table 173: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.7.7 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 Overview) are required to use this feature:
Certificates: *Revoke*

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 174: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.8 Certificate Collections

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

Table 175: Certificate Collections Endpoints

Endpoint	Method	Description	Link
/{id}	GET	Returns the certificate collection with the specified ID.	GET Certificate Collections ID below
/{name}	GET	Returns the certificate collection with the specified name.	GET Certificate Collections Name on page 361
/	GET	Returns all certificate collections with details about the collection configuration.	GET Certificate Collections on page 363
/	POST	Creates a new certificate collection.	POST Certificate Collections on page 365
/	PUT	Updates an existing certificate collection.	PUT Certificate Collections on page 371
/Copy	POST	Creates a new certificate collection based on an existing collection.	POST Certificate Collections Copy on page 374
/{id}/Permissions	POST	Grants the specified collection permissions for the specified role to the specified certificate collection. **Note:** This endpoint will be removed in version 11.	POST Certificate Collections ID Permissions on page 380

2.2.8.1 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 Overview) are required to use this feature:
Certificates: *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.

Table 176: GET CertificateCollections {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 Certificates on page 247) to retrieve a list of all the certificate collections to determine the certificate collection ID.

Table 177: GET CertificateCollections {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
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.8.2 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 Overview) are required to use this feature:
Certificates: *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.

Table 178: GET CertificateCollections 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 247) to retrieve a list of all the certificate collections to determine the certificate collection name. **Tip:** When using the Keyfactor API Endpoint Utility, provide this name without quotation marks.

Table 179: GET CertificateCollections 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
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.8.3 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 Overview) are required to use this feature:
Certificates: *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.

Table 180: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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.
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 181: GET CertificateCollections 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
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.8.4 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 Overview) are required to use this feature:
Certificates: *Read*
Certificate Collections: *Modify*

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 182: 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></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>

Name	In	Description
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 363) 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: <pre>{ "Id": 10, "Name": "Keyexample Collection", "Description": "Certificates in the Keyexample Domain", "Automated": false, "Content": "CN -contains \"keyexample.com\"", "DuplicationField": 2, "ShowOnDashboard": false, "Favorite": true }</pre> 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*, 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

Name	In	Description
		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 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: `{ "Id": 16,`

Name	In	Description
		 `"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` `}` Notice that when Martha runs the command, Gina's goal is achieved.

Table 183: 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:

Value	Description
0	None
1	Common Name
2	Distinguished Name
3	Principal Name

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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.8.5 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.

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Read*
Certificate Collections: *Modify*

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 184: PUT CertificateCollections Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>ID</td><td>Body</td><td>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 363) to locate the ID of the collection you wish to update.</td></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.</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.</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></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>
</table>

Name	In	Description
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*.

Table 185: PUT CertificateCollections 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: **Value** / **Description** 0 / None 1 / Common Name 2 / Distinguished Name 3 / Principal Name
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.8.6 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 Overview) are required to use this feature:
Certificates: *Read*
Certificate Collections: *Modify*

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 186: 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></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>

Name	In	Description
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 363) 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*, 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

Name	In	Description
		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 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: `{` `"Id": 16,`

Name	In	Description
		 `"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` `}` Notice that when Martha runs the command, Gina's goal is achieved.

Table 187: 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:

Value	Description
0	None
1	Common Name
2	Distinguished Name
3	Principal Name

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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.8.7 POST Certificate Collections ID Permissions

The POST /CertificateCollections/{id}/Permissions method is used to set permissions on a certificate collection. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Overview) are required to use this feature:
Security Settings: *Modify*

Warning: When using this method to update an existing certificate collection, all existing RoleId and Permission information must be submitted along with any updates. Any existing permissions that are not included with their full existing data (RoleId and Permission mappings) on an update using this method will be removed from the permissions for the certificate collection. There is not presently a GET method to retrieve the current state of the permissions for certificate collections.

Note: This method has been deprecated and will be removed from the Keyfactor API in release 11. It has been replaced by the endpoint: PUT /Security/Roles/{id}/Permissions/Collection.

Table 188: POST CertificateCollections {id} Permissions Input Parameters

Name	In	Description
id	Path	**Required**. An integer indicating the ID of the certificate collection to update. Use the *GET /CertificateCollections* method (see GET Certificate Collections on page 363) to retrieve a list of all the certificate collections to determine the certificate collection ID.
RoleId	Body	An integer identifying the Keyfactor Command security role that you wish to grant collection security permissions to. Use the *GET /Security/Roles* method (see GET Security Roles on page 914) to retrieve a list of your defined security roles to determine the security role ID to use.
Permissions	Body	An array of the collection permissions that can be granted to the role. Possible values are: • Read • EditMetadata • Recover • Revoke • Delete For example: `"Permissions": [ "Read", "Recover", "Revoke" ]` 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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9 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 189: Certificate Stores Endpoints.

Table 189: Certificate Stores Endpoints

Endpoint	Method	Description	
/	DELETE	Deletes multiple certificate stores specified in the request body.	DELETE Certificate Stores on the next page
/	GET	Returns all certificate stores with paging and option to specify detail level.	GET Certificate Stores on page 384
/	POST	Creates a new certificate store if valid parameters are supplied.	POST Certificate Stores on page 392
/	PUT	Updates an existing certificate store.	PUT Certificate Stores on page 412
/{id}	DELETE	Deletes a certificate store by its GUID.	DELETE Certificate Stores ID on page 432
/{id}	GET	Returns certificate store details for the specified certificate store.	GET Certificate Stores ID on page 432
/{id}/Inventory	GET	Returns certificate inventory for the specified certificate store.	GET Certificate Stores ID Inventory on page 445
/Server (*deprecated)	GET	Returns a list of certificate store servers.	GET Certificate Stores Server on page 447
/Server (*deprecated)	POST	Creates a new certificate store server.	POST Certificate Stores Server on page 449
/Server (*deprecated)	PUT	Updates an existing certificate store server.	PUT Certificate Stores Server on page 454
/Password	PUT	Updates the password for a certificate store.	PUT Certificate Stores Password on page 458
/DiscoveryJob	PUT	Creates a job to find certificate stores.	PUT Certificate Stores Discovery Job on page 461
/AssignContainer	PUT	Assigns a certificate store to a container.	PUT Certificate Stores Assign Container on page 466
/Approve	POST	Approves an array of pending certificate	POST Certificate Stores

Endpoint	Method	Description	
		stores.	Approve on page 474
/Schedule	POST	Creates an inventory schedule for a certificate store.	POST Certificate Stores Schedule on page 482
/Reenrollment	POST	Schedules a reenrollment of a certificate into a certificate store.	POST Certificate Stores Reenrollment on page 485
/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 488
/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 493

2.2.9.1 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 Overview) are required to use this feature:
CertificateStoreManagement: *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 190: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.2 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

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 191: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Certificate Store Search Feature*. 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*.
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 192: 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 496).
ClientMachine	The string value of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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. (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)
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 526 for more information). As of Keyfactor Command v10, this parameter is used to store certificate store server usernames, server passwords, and the UseSSL flag. Built-in certificate stores that typically require configuration of certificate store server parameters include NetScaler and F5 stores. The legacy methods for managing certificate store server credentials have been deprecated but are retained for backwards compatiblity. For more information, see POST Certificate Stores Server

Name	Description
	on page 449. 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 FTP or NetScaler store would contain: `"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"User_Name\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"Password\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` An example server properties parameter POST for an FTP or 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 739): `"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"User_Name\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"Password\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` **Note:** There are three standard properties that are used for any built-in certificate store types that require server credentials (e.g. F5):

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.  **Tip:** Built-in stores that make use of this field include: • AWS stores use this field to store secured versions of the access key and secret. • F5 REST stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl) and primary node information (PrimaryNode, PrimaryNodeCheckRetryWaitSecs, PrimaryNodeCheckRetryMax) and F5 version (F5Version). • F5 SOAP stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • FTP stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • IIS stores (all types) use this field to store the UseSSL flag and the port for WinRM communications. • Java keystores use this field to store type (ProviderType). • NetScaler stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • PEM stores use this field to store the path to the private key file, if defined, and the Boolean value indicating whether a separate private key path is defined.
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	The inventory schedule for this certificate store. The following schedule types are supported: **Name** / **Description** Off — Turn off a previously configured schedule.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>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>
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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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 array 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 key/value pairs for the unique parameters defined</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 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:
<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>"JobProperties": [
{"NetscalerVserver":"MyVirtualServerName"}]</code>
Note: The only built-in certificate store type that makes use of job 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.
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>
</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.3 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 Overview) are required to use this feature:
CertificateStoreManagement: *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 193: 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 496).
ClientMachine	Body	**Required**. The string value of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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. (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)
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 526 for more information). As of Keyfactor Command v10, this parameter is used to store certificate store server usernames, server passwords, and the UseSSL flag. Built-in certificate stores that typically require configuration of certificate store server parameters include NetScaler and F5 stores. The legacy methods for managing certificate store server credentials have been deprecated but are retained for backwards compatiblity. For more information, see POST Certificate Stores Server on page 449.

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 FTP or NetScaler store would contain: `"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"User_Name\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"Password\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` An example server properties parameter POST for an FTP or 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 739): `"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"User_Name\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"Password\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` **Note:** There are three standard properties that are used for any built-in certificate store types that require server credentials (e.g. F5): • ServerUsername

Name	In	Description
		 • ServerPassword • ServerUseSsl These replace the separate certificate store server records that existed in previous versions of Keyfactor Command. For legacy support, if credentials are not provided through store properties during creation or editing of a certificate store, Keyfactor Command will attempt to find a certificate store server record and copy the credentials from it into the store properties for future use.  **Tip:** Built-in stores that make use of this field include: • AWS stores use this field to store secured versions of the access key and secret. • F5 REST stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl) and primary node information (PrimaryNode, PrimaryNodeCheckRetryWaitSecs, PrimaryNodeCheckRetryMax) and F5 version (F5Version). • F5 SOAP stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • FTP stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • IIS stores (all types) use this field to store the UseSSL flag and the port for WinRM communications. • Java keystores use this field to store type (ProviderType). • NetScaler stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • PEM stores use this field to store the path to the private key file, if defined, and the Boolean value indicating whether a separate private key path is defined.
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	The inventory schedule for this certificate store. 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>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>
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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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 array 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 key/value pairs for the unique parameters defined</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 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:
<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>"JobProperties": [
{"NetscalerVserver":"MyVirtualServerName"}
]</code>
Note: The only built-in certificate store type that makes use of job 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.
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>
</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 array 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 NetScaler and F5 devices and set at the server level, not the certificate store level (see POST Certificate Stores Server on page 449).</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
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 723 for more information.

The possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Value</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": {
   "Value": {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": {
   "Value": "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, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>ProviderType Para-</td><td>An array containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / **Description** meterValue-s **Name** / **Description** Id — The Keyfactor Command reference ID for the PAM provider type parameter. Value — 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 — The Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used. InstanceGuid — 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 array containing information about the provider. PAM provider details include: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID for the PAM provider. Name — A string indicating the internal name for the PAM provider. Area — An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indicating they are used for certificate stores. ProviderType — An array containing details about the provider type for the provider, including:

Name	In	Description
		Name / **Description** **Name** / **Description** **Name** / **Description** Provider-Type: An array containing the values for **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. Provider Type Params: An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. See below instance of *ProviderTypeParam* for details.

Name	In	Description
		Name / **Description** **Name** / **Description** **Name** / **Description** Para-mValues — the provider types specified by ProviderTypeParams. See the previous level of *Provider-TypeParamValues* for details. SecuredAre-aId — An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any. You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting passwords for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers.

Name	In	Description
		Name / **Description** **Name**: Provider-Type Param **Description**: An array of parameters 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 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

Name	In	Description
		Name / **Description** **Name** / **Description** **Name** / **Description** 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 739. Provider-Type: An array containing details for the provider type. **Name** / **Description-tion** Id: A string indicating the Keyfactor Command reference GUID for the PAM provider type parameter. Name: A string indicating the internal name for the PAM

<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>Descrip-tion</th></tr>
<tr><td></td><td>provider type para-meter.</td></tr>
<tr><td>Provider-TypeParams</td><td>Unused field</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>

Table 194: 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 496).
ClientMachine	The string value of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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. (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)
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 526 for more information). As of Keyfactor Command v10, this parameter is used to store certificate store server usernames, server passwords, and the UseSSL flag. Built-in certificate stores that typically require configuration of certificate store server parameters include NetScaler and F5 stores. The legacy methods for managing certificate store server credentials have been deprecated but are retained for backwards compatiblity. For more information, see POST Certificate Stores Server

Name	Description
	on page 449. 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 FTP or NetScaler store would contain: `"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"User_Name\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"Password\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` An example server properties parameter POST for an FTP or 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 739): `"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"User_Name\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"Password\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` **Note:** There are three standard properties that are used for any built-in certificate store types that require server credentials (e.g. F5):

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.  **Tip:** Built-in stores that make use of this field include: • AWS stores use this field to store secured versions of the access key and secret. • F5 REST stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl) and primary node information (PrimaryNode, PrimaryNodeCheckRetryWaitSecs, PrimaryNodeCheckRetryMax) and F5 version (F5Version). • F5 SOAP stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • FTP stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • IIS stores (all types) use this field to store the UseSSL flag and the port for WinRM communications. • Java keystores use this field to store type (ProviderType). • NetScaler stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • PEM stores use this field to store the path to the private key file, if defined, and the Boolean value indicating whether a separate private key path is defined.
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	The inventory schedule for this certificate store. The following schedule types are supported: **Name** / **Description** Off — Turn off a previously configured schedule.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>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>
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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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 array 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 key/value pairs for the unique parameters defined</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 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:
<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>"JobProperties": [
{"NetscalerVserver":"MyVirtualServerName"}]</code>
Note: The only built-in certificate store type that makes use of job 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.
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>
</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.4 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 Overview) are required to use this feature:
CertificateStoreManagement: *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.

Warning: 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 195: 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 496).
ClientMachine	Body	**Required**. The string value of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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. (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)
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 526 for more information). As of Keyfactor Command v10, this parameter is used to store certificate store server usernames, server passwords, and the UseSSL flag. Built-in certificate stores that typically require configuration of certificate store server parameters include NetScaler and F5 stores. The

Name	In	Description
		legacy methods for managing certificate store server credentials have been deprecated but are retained for backwards compatiblity. For more information, see POST Certificate Stores Server on page 449. 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 FTP or NetScaler store would contain: `"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"User_Name\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"Password\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` An example server properties parameter POST for an FTP or 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 739): `"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"User_Name\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"Password\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"`

Name	In	Description
		Note: There are three standard properties that are used for any built-in 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. **Tip:** Built-in stores that make use of this field include: • AWS stores use this field to store secured versions of the access key and secret. • F5 REST stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl) and primary node information (PrimaryNode, PrimaryNodeCheckRetryWaitSecs, PrimaryNodeCheckRetryMax) and F5 version (F5Version). • F5 SOAP stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • FTP stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • IIS stores (all types) use this field to store the UseSSL flag and the port for WinRM communications. • Java keystores use this field to store type (ProviderType). • NetScaler stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl). • PEM stores use this field to store the path to the private key file, if defined, and the Boolean value indicating whether a separate private key path is defined.
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.
Invent-	Bod-	The inventory schedule for this certificate store. The following schedule types are supported:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>orySchedule</td><td>y</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>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>
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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>

Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>

Note: Although the 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 array 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 key/value pairs for the unique parameters defined</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 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:
<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>"JobProperties": [
{"NetscalerVserver":"MyVirtualServerName"}
]</code>
Note: The only built-in certificate store type that makes use of job 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.
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>
</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 array 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 NetScaler and F5 devices and set at the server level, not the certificate store level (see POST Certificate Stores Server on page 449).</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
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 723 for more information.

The possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Value</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": {
   "Value": {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": {
   "Value": "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, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>ProviderType Para-</td><td>An array containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / **Description** meterValue-s **Name** / **Description** Id — The Keyfactor Command reference ID for the PAM provider type parameter. Value — 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 — The Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used. InstanceGuid — 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 array containing information about the provider. PAM provider details include: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID for the PAM provider. Name — A string indicating the internal name for the PAM provider. Area — An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indicating they are used for certificate stores. ProviderType — An array containing details about the provider type for the provider, including:

Name	In	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. Provider Type Params — An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. See below instance of *ProviderTypeParam* for details. Provider-Type — An array containing the values for

Name	In	Description
		Name / **Description** **Name** / **Description** **Name** / **Description** Para-mValues — the provider types specified by ProviderTypeParams. See the previous level of *Provider-TypeParamValues* for details. SecuredAre-aId — An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any. You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting passwords for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers.

Name	In	Description
		Name / **Description** **Name**: Provider-Type Param **Description**: An array of parameters 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 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

Name	In	Description
		Name / **Description** **Name** / **Description** **Name** / **Description** 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 739. Provider-Type: An array containing details for the provider type. **Name** / **Descrip-tion** Id: A string indicating the Keyfactor Command reference GUID for the PAM provider type para-meter. Name: A string indicating the internal name for the PAM

<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>Descrip-tion</th></tr>
<tr><td></td><td>provider type para-meter.</td></tr>
<tr><td>Provider-TypeParams</td><td>Unused field</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>

Table 196: 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 496).
ClientMachine	The string value of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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. (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)
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 526 for more information). As of Keyfactor Command v10, this parameter is used to store certificate store server usernames, server passwords, and the UseSSL flag. Built-in certificate stores that typically require configuration of certificate store server parameters include NetScaler and F5 stores. The legacy methods for managing certificate store server credentials have been deprecated but are retained for backwards compatiblity. For more information, see POST Certificate Stores Server

Name	Description
	on page 449. 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 FTP or NetScaler store would contain: `"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"User_Name\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"Password\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` An example server properties parameter POST for an FTP or 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 739): `"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"User_Name\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"Password\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` **Note:** There are three standard properties that are used for any built-in certificate store types that require server credentials (e.g. F5):

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


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.

Tip: Built-in stores that make use of this field include:

AWS stores use this field to store secured versions of the access key and secret.
F5 REST stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl) and primary node information (PrimaryNode, PrimaryNodeCheckRetryWaitSecs, PrimaryNodeCheckRetryMax) and F5 version (F5Version).
F5 SOAP stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
FTP stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
IIS stores (all types) use this field to store the UseSSL flag and the port for WinRM communications.
Java keystores use this field to store type (ProviderType).
NetScaler stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
PEM stores use this field to store the path to the private key file, if defined, and the Boolean value indicating whether a separate private key path is defined.

</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>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>
</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>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>
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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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 array 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 key/value pairs for the unique parameters defined</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 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:
<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>"JobProperties": [
{"NetscalerVserver":"MyVirtualServerName"}]</code>
Note: The only built-in certificate store type that makes use of job 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.
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>
</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.5 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 Overview) are required to use this feature:
CertificateStoreManagement: *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 197: 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 384) to retrieve a list of all the certificate stores to determine the certificate store GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.6 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

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 198: GET Certificate Stores {id} Input Parameters

Name	In	Description
id	Path	**Required**. A string indicating the GUID of the certificate store within Keyfactor Command.

Table 199: 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 496).
ClientMachine	The string value of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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. (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)
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 526 for more information). As of Keyfactor Command v10, this parameter is used to store certificate store server usernames, server passwords, and the UseSSL flag. Built-in certificate stores that typically require configuration of certificate store server parameters include NetScaler and F5 stores. The legacy methods for managing certificate store server credentials have been deprecated but are retained for backwards compatiblity. For more information, see POST Certificate Stores Server on page 449. When reading this field, the values are returned as simple key value pairs, with the values being indi-

Name	Description
	vidual 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 FTP or NetScaler store would contain: `"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"User_Name\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"Password\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` An example server properties parameter POST for an FTP or 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 739): `"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"User_Name\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"Password\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` **Note:** There are three standard properties that are used for any built-in certificate store types that require server credentials (e.g. F5): • ServerUsername • ServerPassword

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

Tip: Built-in stores that make use of this field include:

AWS stores use this field to store secured versions of the access key and secret.
F5 REST stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl) and primary node information (PrimaryNode, PrimaryNodeCheckRetryWaitSecs, PrimaryNodeCheckRetryMax) and F5 version (F5Version).
F5 SOAP stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
FTP stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
IIS stores (all types) use this field to store the UseSSL flag and the port for WinRM communications.
Java keystores use this field to store type (ProviderType).
NetScaler stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
PEM stores use this field to store the path to the private key file, if defined, and the Boolean value indicating whether a separate private key path is defined.

</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>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).</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 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>
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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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 array 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 key/value pairs for 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</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>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:
<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>"JobProperties": [
{"NetscalerVserver":"MyVirtualServerName"}]</code>
Note: The only built-in certificate store type that makes use of job 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.
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>
</table>
</td></tr>
<tr><td>SetNewPassword-Allowed</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 array 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 NetScaler and F5 devices and set at the server level, not the certificate store level (see <a>POST Certificate Stores Server on page 449).
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).</td></tr>
</table>

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

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 723 for more information.

The possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Value</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": {
   "Value": {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": {
   "Value": "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, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>ProviderTypeParameterValues</td><td>An array containing the values for the PAM provider types specified by ProviderTypeParams. The provider type parameter values include:</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description** **Name** / **Description** Id — The Keyfactor Command reference ID for the PAM provider type parameter. Value — 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 — The Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used. InstanceG-uid — 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 array containing information about the provider. PAM provider details include: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID for the PAM provider. Name — A string indicating the internal name for the PAM provider. Area — An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indicating they are used for certificate stores. ProviderType — An array containing details about the provider type for the provider, including:

Name	Description
	Name / **Description** **Name** / **Description** **Name** / **Description** **Name** / **Description** Id — A string indicating the Keyfactor Command reference GUID for the provider type. Name — A string that indicates the name of the provider type. ProviderTypeParams — An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. See below instance of *ProviderTypeParam* for details. ProviderTypeParamValues — An array containing the values for the provider types specified by ProviderTypeParams. See the previous level of *ProviderTypeParamValues* for details.

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. You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting passwords for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers. ProviderTypeParam: An array of parameters 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. 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 739. Provider- — An array containing details for the

Name	Description
	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 in the Keyfactor secrets table (false). This value is automatically set by Keyfactor Command.

Name	Description
	Note: Secret data is stored in the secrets table or a PAM provider and is not returned in responses.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.7 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

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 200: 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 201: 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 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>The date, in UTC, on which the certificate was issued by the certificate authority.</td></tr>
<tr><td>NotAfter</td><td>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 array of entry parameters associated with the certificate 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 531 for more information about entry parameters.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.8 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, FTP, NetScaler, and any 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

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 certificate store (see GET Certificate Stores on page 384).

Table 202: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Certificate Store Search Feature* section. 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.
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 203: GET Certificate Stores Server Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>The ID of the server.</td></tr>
<tr><td>Username</td><td>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.</td></tr>
<tr><td>Password</td><td>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.</td></tr>
<tr><td>UseSSL</td><td>A Boolean that indicates whether Keyfactor Command will use SSL to communicate with the server (true) or not (false).</td></tr>
<tr><td>ServerType</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</td></tr>
<tr><td>1</td><td>NetScaler</td></tr>
<tr><td>2</td><td>FTP</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>
</td></tr>
<tr><td>Name</td><td>The host name of the server.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.9 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 Overview) are required to use this feature:
CertificateStoreManagement: *Modify*

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 certificate store (see POST Certificate Stores on page 392). The Management Portal has additional functionality, such as being able to set different credentials for different stores on the same server, which use the new API endpoint.

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 204: 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>Required. The username used to connect to the certificate store. Username parameters include:
<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 739) 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 739) 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 GET PAM Providers on page 739) 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 GET PAM Providers on page 739) 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>
<tr><td>ServerType</td><td>Body</td><td>An integer indicating the type of server. Possible values include (plus any custom values):</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>0</td><td>F5 Web Server & F5 SSL Profiles</td></tr>
<tr><td>1</td><td>NetScaler</td></tr>
<tr><td>2</td><td>FTP</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 GET Certificate Store Types on page 526) 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. This value must be specified if you are using PAM to store your username and/or password and your PAM provider has been configured to be linked to a specific certificate store container.</td></tr>
</table>

Table 205: 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 1 / NetScaler 2 / FTP 3 / F5 Web Server REST 4 / F5 SSL Profiles REST 5 / F5 CA Bundles REST
Name	The host name of the server.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.10 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 Overview) are required to use this feature:
CertificateStoreManagement: *Modify*

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. The Management Portal has additional functionality, such as being able to set different credentials for different stores on the same server, which use the new PUT Certificate Stores on page 412 API endpoint. Using this deprecated API endpoint could potentially, for instance, overwrite all cert stores on the server.

Warning: 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 206: PUT Certificate Stores Server Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td>Body</td><td>The ID of the server.</td></tr>
<tr><td>Username</td><td>Body</td><td>Required. The username used to connect to the certificate store. Username parameters
<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 739) 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 739) 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></td><td></td><td></td></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 GET PAM Providers on page 739) 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 GET PAM Providers on page 739) 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>
<tr><td>Name</td><td>Body</td><td>Required. The host name of the server.</td></tr>
</table>

Name	In	Description
Container	Body	An integer that identifies the certificate store container into which the certificate store should be placed for organizational and management purposes. This value must be specified if you are using PAM to store your username and/or password and your PAM provider has been configured to be linked to a specific certificate store container.

Table 207: 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: 0 — Description: F5 Web Server & F5 SSL Profiles Value: 1 — Description: NetScaler Value: 2 — Description: FTP Value: 3 — Description: F5 Web Server REST Value: 4 — Description: F5 SSL Profiles REST Value: 5 — Description: F5 CA Bundles REST
Name	The host name of the server.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.11 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 Overview) are required to use this feature:
CertificateStoreManagement: *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 208: PUT Certificate Stores Password Input Parameters

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CertStoreID</td><td>Body</td><td>Required. A string indicating the GUID of the certificate store. Use the GET CertificateStores method (see GET Certificate Stores on page 384) to retrieve a list of all your certificate stores to determine the GUID of the store.</td></tr>
<tr><td>NewPassword</td><td>Body</td><td>Required. A array 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:
<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 739) 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.</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 GET PAM Providers on page 739) 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>"NewPassword": {
 "Provider": 2,
 "Parameters": {
  "SecretId": 5
 }
},</pre>For CyberArk, this might be:
<pre>"NewPassword": {
 "Provider": 5,
 "Parameters": {
  "Folder": "Root",
  "Object": "F5Password"</pre></td></tr>
</table></td></tr>
</table>

Name	Type	Description
		Name / **Description** } }, For a password stored in the Keyfactor Command database, this might be: `"NewPassword": { "SecretValue": "P@ssw0rd" }`

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.12 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.
- F5 bundle and SSL certificates discovered by the Keyfactor Windows Orchestrator on F5 devices using the F5 REST API (v14 and up).
- F5 bundle and SSL certificates discovered by the Keyfactor Universal Orchestrator with a custom extension to support F5. 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 474).

Tip: The following permissions (see Security Overview) are required to use this feature:
CertificateStoreManagement: *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 209: 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. (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) The default is 0 for a JKS discovery.
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. 2021-05-19T16:23:01Z).
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 example, search for files with the extension "jks" in order to exclude files with other extensions such as "txt". The dot should not be included when specifying extensions.

Name	In	Description
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>Required*. The username used to connect to the certificate store server.

<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 739) 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 739) 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></td><td></td><td>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 include F5, FTP, NetScaler, and any custom method you've defined to support this.</td></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 GET PAM Providers on page 739) 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 GET PAM Providers on page 739) 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",</pre></td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><pre>"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 include F5, FTP, NetScaler, 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.13 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 Overview) are required to use this feature:
CertificateStoreManagement: *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 210: 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 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 211: 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 496).
ClientMachine	The string value of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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. (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)
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 526 for more information). As of Keyfactor Command v10, this parameter is used to store certificate store server usernames, server passwords, and the UseSSL flag. Built-in certificate stores that typically require configuration of certificate store server parameters include NetScaler and F5 stores. The legacy methods for managing certificate store server credentials have been deprecated but are retained for backwards compatiblity. For more information, see POST Certificate Stores Server

Name	Description
	on page 449. 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 FTP or NetScaler store would contain: `"{ \"ServerUsername\":{\"value\":{\"SecretValue\":\"User_Name\"}}, \"ServerPassword\":{\"value\":{\"SecretValue\":\"Password\"}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` An example server properties parameter POST for an FTP or 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 739): `"{ \"ServerUsername\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"User_Name\"}}}, \"ServerPassword\":{\"value\":{\"Provider\":\"1\",\"Parameters\": {\"SecretId\":\"Password\"}}}, \"ServerUseSsl\":{\"value\":\"true\"} }"` **Note:** There are three standard properties that are used for any built-in certificate store types that require server credentials (e.g. F5):

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


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.

Tip: Built-in stores that make use of this field include:

AWS stores use this field to store secured versions of the access key and secret.
F5 REST stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl) and primary node information (PrimaryNode, PrimaryNodeCheckRetryWaitSecs, PrimaryNodeCheckRetryMax) and F5 version (F5Version).
F5 SOAP stores (all types) use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
FTP stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
IIS stores (all types) use this field to store the UseSSL flag and the port for WinRM communications.
Java keystores use this field to store type (ProviderType).
NetScaler stores use this field to store secured versions of the server authentication information (ServerUsername, ServerPassword, ServerUseSsl).
PEM stores use this field to store the path to the private key file, if defined, and the Boolean value indicating whether a separate private key path is defined.

</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>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>
</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>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>

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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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 array 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 key/value pairs for the unique parameters defined</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 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:
<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>"JobProperties": [
{"NetscalerVserver":"MyVirtualServerName"}]</code>
Note: The only built-in certificate store type that makes use of job 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.
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>
</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.14 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 461). 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 Overview) are required to use this feature:
CertificateStoreManagement: *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 212: 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 384) 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 496) 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. (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)
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 526 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\"} }"` This field is **required** for certificate store types that store additional properties in this parameter.
Password	Body	**Required**. An array 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 NetScaler and F5 devices and set at the server level, not the certificate store level (see POST Certificate Stores Server on page 449). Certificate stores that require credentials support up to three possible credential options:

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

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 723 for more information.

The possible values are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Value</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": {
   "Value": {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": {
   "Value": "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, this will be used. This value is automatically set by Keyfactor Command.</td></tr>
<tr><td>Provider-TypePara-meterValues</td><td>An array containing the values for the provider types specified by ProviderTypeParams. PAM provider type parameter values include:</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / **Description** Id — The Keyfactor Command reference ID for the PAM provider type parameter. Value — 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 — The Keyfactor Command reference ID for the PAM provider. If you are attaching to something with an integer Id, this will be used. InstanceGuid — 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 array containing information about the provider. PAM provider details include: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID for the PAM provider. Name — A string indicating the internal name for the PAM provider. Area — An integer indicating the area of Keyfactor Command the provider is used for. PAM providers generally have a value of 1, indicating they are used for certificate stores. ProviderType — An array containing details about the provider type for the provider, including:

Name	In	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 *ProviderTypeParam* for details. ProviderType — An array containing the values

Name	In	Description
		Name / **Description** **Name** / **Description** **Name** / **Description** Para-mValues — for the provider types specified by ProviderTypeParams. See the previous level of *Provider-TypeParamValues* for details. SecuredAre-aId — An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any. You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting passwords for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and

Name	In	Description
		Name / **Description** F5) as long as they were not in containers.
		Provider-Type Param An array of parameters 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 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:

Name	In	Description
		Name / **Description** **Name** / **Description** **Name** / **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 certificate store to use the PAM provider (true). For an example, see GET PAM Providers on page 739. ProviderType: An array 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

<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>Descrip-tion</th></tr>
<tr><td></td><td>indicating the internal name for the PAM provider type para-meter.</td></tr>
<tr><td>Provider-TypeParams</td><td>Unused field</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Provider</td><td>An integer indicating the Keyfactor Command reference ID for the PAM provider.</td></tr>
<tr><td>IsManaged</td><td>A Boolean indicating whether the credentials for the store are managed by a PAM provider (true) or stored in the Keyfactor secrets table (false). This value is automatically set by Keyfactor Command.</td></tr>
</table>
This field is required for Java keystores.
</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.15 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 Overview) are required to use this feature:
CertificateStoreManagement: *Schedule*

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 213: POST Certificate Stores Schedule Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>StoreIds</td><td>Body</td><td>Required. An array of strings providing the certificate store GUIDs to schedule.</td></tr>
<tr><td>Schedule</td><td>Body</td><td>Required. The inventory schedule for the certificate store(s). 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>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. 2021-05-19T16:23:01Z).</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": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.16 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 700).

Tip: The following permissions (see Security Overview) are required to use this feature:
CertificateEnrollment: *EnrollCSR*
CertificateStoreManagement: *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.

In addition, the 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 214: 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 384) 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 384) 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 array of key/value pairs for 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-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*

Name	In	Description
		application setting will be used (see Application Settings: Agents Tab in the *Keyfactor Command Reference Guide*).

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.17 POST Certificate Stores Certificates Add

The POST /CertificateStores/Certificates/Add method is used to add a certificate to one or more certificate stores. The POST request must contain a certificate ID and an array of certificate store GUIDs that identify the stores to which the certificate should be added. 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 700).

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Read*
CertificateStoreManagement: *Schedule*

Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. See *Certificate Permissions* and *Container Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection and container permissions.

Table 215: 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>CertificateStores</td><td>Body</td><td>Required. An array of 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>CertificateStoreIds</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 array of key/value pairs 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 overwritten with the certificate being added (true) or not (false). The default is false.
Use the GET /Certificates/Locations/{id} method (see GET Certificates Locations ID on page 224) 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>EntryPassword</td><td>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</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>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 739) 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 GET PAM Providers on page 739) 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></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>},
For CyberArk, this might be:
<pre>"EntryPassword": {
  "Provider": 5,
  "Parameters": {
    "Folder":
"Root",
    "Object":
"F5Password"
  }
},</pre></td></tr>
</table>
</td></tr>
<tr><td>PfxPassword</td><td>A string that sets the password to use when saving a certificate with its private key in the certificate store. This is only relevant if there's a 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. The inventory schedule for the add job. Possible schedule values 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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.9.18 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 700).

Tip: The following permissions (see Security Overview) are required to use this feature:
Certificates: *Read*
CertificateStoreManagement: *Schedule*

Permissions for certificates and certificate stores can be set at either the global or certificate collection and certificate store container level. See *Certificate Permissions* and *Container Permissions* in the *Keyfactor Command Reference Guide* for more information about global vs collection and container permissions.

Table 216: 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 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 232) 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 array of key/value pairs 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-c4ad4569b4e7"
   }
]
</pre></td></tr>
<tr><td>Schedule</td><td>Body</td><td>Required. 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>
</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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
Note: Although the 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.10 Certificate Store Containers

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

Table 217: Certificate Store Containers Endpoints

Endpoint	Method	Description	Link
/	GET	Returns a list of certificate store containers.	GET Certificate Store Containers below
/	POST	Adds a certificate store container.	POST Certificate Store Containers on page 499
/{id}	DELETE	Deletes a certificate store container.	DELETE Certificate Store Containers ID on page 507
/{id}	GET	Returns details for the specified certificate store container.	GET Certificate Store Containers ID on page 508
/{id}	PUT	Edits a certificate store container.	PUT Certificate Store Containers on page 503

2.2.10.1 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

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 218: GET Certificate Store Containers 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Containers Search Feature*. 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.
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 219: GET Certificate Stores Containers Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the container.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the container.</td></tr>
<tr><td>Over-writeSchedules</td><td>A Boolean indicating whether the schedule set on the container will overwrite schedules set individually on the certificate stores (true) or not (false).</td></tr>
<tr><td>Schedule</td><td>A string containing the inventory schedule set for the container. 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table></td></tr>
</table>

Name	Description
	Note: Although the 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	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. (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)
StoreCount	An integer indicating the number of stores of the type referenced by CertStoreType in the container.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.10.2 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 Overview) are required to use this feature:
CertificateStoreManagement: *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 220: 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>A string containing the inventory schedule set for the container. 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other schedules, only the schedules shown here—that are available in the Management</td></tr>
</table>

Name	In	Description
		Portal for this functionality—are valid for this endpoint.
CertStoreTyp-e	Body	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. (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) The default is 0 for a JKS keystore.

Table 221: POST Certificate Stores Containers Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the container.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the container.</td></tr>
<tr><td>Schedule</td><td>A string containing the inventory schedule set for the container. 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other schedules,</td></tr>
</table>

Name	Description
	only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. (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)

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.10.3 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 Overview) are required to use this feature:
CertificateStoreManagement: *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.

Warning: 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 222: 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>A string containing the inventory schedule set for the container. 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Note: Although the 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.
CertStoreTyp-e	Body	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. (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) The default is 0 for a JKS keystore.

Table 223: PUT Certificate Store Containers Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the container.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the container.</td></tr>
<tr><td>Schedule</td><td>A string containing the inventory schedule set for the container. 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other schedules,</td></tr>
</table>

Name	Description
	only the schedules shown here—that are available in the Management Portal for this functionality—are valid for this endpoint.
CertStoreType	An integer indicating the ID of the certificate store type, as defined in Keyfactor Command, for this certificate store. (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)

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.10.4 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 Overview) are required to use this feature:
CertificateStoreManagement: *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 224: 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 496) to retrieve a list of all the certificate store containers to determine the certificate store container ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.10.5 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

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 225: 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 496) to retrieve a list of all the certificate store containers to determine the certificate store container ID.

Table 226: GET Certificate Stores Containers {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the container.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the container.</td></tr>
<tr><td>Schedule</td><td>A string containing the inventory schedule set for the container. 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the Swagger Example Value may show examples of various other schedules,</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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. (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)</td></tr>
<tr><td>CertificateStores</td><td>An array of 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>The string value of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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 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>CertStoreInventoryJobId</td><td>A string indicating the GUID that identifies the inventory job for the certificate store in the Keyfactor Command database. This will be null if an inventory schedule is not set for the certificate store.</td></tr>
</table></td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></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. (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)</td></tr><tr><td>Approved</td><td>A Boolean that indicates whether a certificate store is approved (true) or not (false). If a certificate store is approved, it can be used and updated. A certificate store that has been discovered using the discover feature but not yet marked as approved will be false here.</td></tr><tr><td>CreateIfMissing</td><td>A Boolean that indicates whether a new certificate store should be created with the information provided (true) or not (false). This option is only valid for Java keystores and any custom certificate store types you have defined to support this functionality.</td></tr><tr><td>Properties</td><td>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 526 for more information). When reading this field, the values are returned as simple key value pairs, with the values being individual values. When writing, the values are specified as objects, though they are typically single values. For example, on a GET request for a PEM store configured with a separate private key, the contents of this field might be: <pre>"{ \"privateKeyPath\":\"/opt/app/mystore.key\", \"separatePrivateKey\":\"true\" }"</pre> 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:</td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td><pre>"{ \"privateKeyPath\":{\"value\":\"/- opt/app/mystore.key\"}, \"separatePrivateKey\":{\"value\":\"true\"} }"</pre> **Tip:** Built-in stores that make use of this field include: • AWS stores use this field to store secured versions of the access key and secret. • F5 REST stores (all types) use this field to store the primary node information (PrimaryNode, PrimaryNodeCheckRetryWaitSecs, PrimaryNodeCheckRetryMax) and F5 version (F5Version). • IIS stores (all types) use this field to store the port for WinRM communications. • PEM stores use this field to store the path to the private key file, if defined, and the Boolean value indicating whether a separate private key path is defined.</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>The inventory schedule for this certificate store.</td></tr><tr><td>ReenrollmentStatus</td><td>An array that indicates whether the certificate store can use the re-enrollment function with accompanying data about the re-enrollment job.</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 array indicating the source for and details of the credential</td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>information Keyfactor Command will use to access the certificates in a specific certificate store (the store password). **Note:** Secret data is stored in the secrets table or a PAM provider and is not returned in responses.</td></tr></table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.11 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 227: 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 519
/	DELETE	Delete multiple certificate store types using StoreType number.	DELETE Certificate Store Types on page 525
/	GET	Returns all certificate store types with paging and options to the specified detail level.	GET Certificate Store Types on page 526
/	POST	Creates a new certificate store type.	POST Certificate Store Types on page 531
/	PUT	Updates a certificate store type using StoreType number.	PUT Certificate Store Types on page 543

2.2.11.1 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 Overview) are required to use this feature:
CertificateStoreManagement: *Modify*

Table 228: 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 526) to retrieve a list of all the certificate store types to determine the certificate store type ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.11.2 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

Table 229: 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 526) to retrieve a list of all the certificate store types to determine the certificate store type ID.

Table 230: 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	A unique integer 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 array containing a series of Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of 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 • MultipleChoice • Secret

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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 any built-in 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>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).</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 authen-

</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>ticated 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>An array 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 user-name 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>
<tr><td>CustomAliasAllowed</td><td>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).</td></tr>
<tr><td>EntryParameters</td><td>An array of unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr><tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr><tr><td>Type</td><td>A string containing the type of the entry parameter: - String - Bool - MultipleChoice - Secret </td></tr><tr><td>RequiredWhen</td><td>An array of Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: - HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. - OnAdd: If set to true, a value must be provided for this field when configuring an add 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>

Name	Description
	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).
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.11.3 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

Table 231: GET Certificate Store Types Name {ShortName} 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 526) to retrieve a list of all the certificate store types to determine the certificate store type short name.

Table 232: GET Certificate Store Types Name {ShortName} 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	A unique integer 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 array containing a series of Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of 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 • MultipleChoice • Secret

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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 any built-in 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>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).</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 authen-

</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>ticated 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>An array 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 user-name 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>
<tr><td>CustomAliasAllowed</td><td>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 certi-ficate (e.g. IIS trusted root).</td></tr>
<tr><td>EntryParameters</td><td>An array of unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr><tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr><tr><td>Type</td><td>A string containing the type of the entry parameter: - String - Bool - MultipleChoice - Secret </td></tr><tr><td>RequiredWhen</td><td>An array of Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: - HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. - OnAdd: If set to true, a value must be provided for this field when configuring an add 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>

Name	Description
	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).
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.11.4 DELETE Certificate Store Types

The DELETE /CertificateStoreTypes method is used to delete multiple certificate store types in one request. The certificate store type IDs should be supplied in the request body as a JSON array of integers. 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 Overview) are required to use this feature:
CertificateStoreManagement: *Modify*

Table 233: DELETE Certificate Store Types Input Parameters

Name	In	Description
ids	Body	**Required**. An array of 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.11.5 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 Overview) are required to use this feature:
CertificateStoreManagement: *Read*

Table 234: 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 235: 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	A unique integer 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 array containing a series of Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of 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 • MultipleChoice • Secret

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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 any built-in 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>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).</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 authen-

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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>ticated 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*.</td></tr></table>
StorePathValue	An array containing the value(s) for the certificate store path.
PrivateKeyAllowed	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).
ServerRequired	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 user-name and password to connect to the remote server.
PowerShell	A Boolean that indicates whether jobs for the store type are implemented using PowerShell (true) instead of a .NET class or not (false).
BlueprintAllowed	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*.
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 unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr><tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr><tr><td>Type</td><td>A string containing the type of the entry parameter: - String - Bool - MultipleChoice - Secret </td></tr><tr><td>RequiredWhen</td><td>An array of Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: - HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. - OnAdd: If set to true, a value must be provided for this field when configuring an add 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>

Name	Description
	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).
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.11.6 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 Overview) are required to use this feature:
CertificateStoreManagement: *Modify*

Table 236: 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 array containing a series of 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 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 of the property: • String • Bool • MultipleChoice

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>• Secret If you choose to define a property, this field is **required**.</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 any built-in 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: `"Properties": [` `{` `"Name": "Pets",` `"DisplayName": "Popular Pets",` `"Type": "MultipleChoice",` `"DependsOn": "",` `"DefaultValue": "Cat,Dog,Fish,Rat,Mouse",`

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

<pre>
      "Required": false
   }
]
</pre>

This value is unset by default.</td></tr>
<tr><td>PasswordOptions</td><td>Body</td><td>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:
• 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.

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	An array 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 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 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**.</td></tr><tr><td>RequiredWhen</td><td>An array of 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</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>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:
<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
</td></tr>
</table>

Name	In	Description
		 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 237: 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	A unique integer 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 array containing a series of Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of 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 • MultipleChoice • Secret

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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 any built-in 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>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).</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 authen-

</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>ticated 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>An array 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 user-name 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>
<tr><td>CustomAliasAllowed</td><td>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 certi-ficate (e.g. IIS trusted root).</td></tr>
<tr><td>EntryParameters</td><td>An array of unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr><tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr><tr><td>Type</td><td>A string containing the type of the entry parameter: - String - Bool - MultipleChoice - Secret </td></tr><tr><td>RequiredWhen</td><td>An array of Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: - HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. - OnAdd: If set to true, a value must be provided for this field when configuring an add 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>

Name	Description
	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).
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.11.7 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 Overview) are required to use this feature:
CertificateStoreManagement: *Modify*

Warning: 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 238: PUT Certificate Store Types Input Parameters

Name	In	Description
StoreType	Body	**Required**. 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 array containing a series of 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 unique parameters for the certificate store type. In the Keyfactor Command Management Portal these are known as *Custom Fields*. Property parameters 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

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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**.</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 any built-in 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

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

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>the*Keyfactor Command Reference Guide*. The default value is *Default*.</td></tr></table>
StorePathType	Body	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.
StorePathValue	Body	An array 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 Blue-*

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>prints in the Keyfactor Command Reference Guide. The default is false.</td></tr>
<tr><td>CustomAliasAllowed</td><td>Body</td><td>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.</td></tr>
<tr><td>EntryParameters</td><td>Body</td><td>An array of 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.</td></tr><tr><td>RequiredWhen</td><td>An array of Boolean values indicating the circum-</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>stances 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 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:
</td></tr>
</table>

Name	In	Description
		<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 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 239: 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	A unique integer 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 array containing a series of Boolean values that indicate whether the certificate store type is enabled for the following functions: • Add • Create • Discovery • Enrollment • Remove
Properties	An array of 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 • MultipleChoice • Secret

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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 any built-in 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>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).</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 authen-</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>ticated 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>An array 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 user-name 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>
<tr><td>CustomAliasAllowed</td><td>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 certi-ficate (e.g. IIS trusted root).</td></tr>
<tr><td>EntryParameters</td><td>An array of unique parameters that are required when performing management jobs on a certificate store of this type. Entry parameter options include:</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Name</td><td>A string containing the short name of the entry parameter.</td></tr><tr><td>DisplayName</td><td>A string containing the full display name of the entry parameter.</td></tr><tr><td>Type</td><td>A string containing the type of the entry parameter: - String - Bool - MultipleChoice - Secret </td></tr><tr><td>RequiredWhen</td><td>An array of Boolean values indicating the circumstances under which a value is required to be provided for this entry parameter. These are: - HasPrivateKey: If set to true, a value must be provided for this field when configuring a management job (either add or remove) if the certificate has an associated private key in Keyfactor Command. This would be the case, for example, when doing a PFX enrollment and adding the resulting certificate to a certificate store. - OnAdd: If set to true, a value must be provided for this field when configuring an add 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>

Name	Description
	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).
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.12 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 240: 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 below
/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 559
/Generate	POST	Generate and configure a CSR request.	POST CSR Generation Generate on page 560

2.2.12.1 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 Overview) are required to use this feature:
CertificateEnrollment: *PendingCsr*

Table 241: 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 559) to retrieve a list of all the pending CSRs to determine the CSR IDs.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.12.2 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 page 559).

Tip: The following permissions (see Security Overview) are required to use this feature:
CertificateEnrollment: *PendingCsr*

Table 242: GET CSR Generation Pending {id} Input Parameters

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

Table 243: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.12.3 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. The IDs should be supplied in the request body as a JSON array of integers. 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 Overview) are required to use this feature:
CertificateEnrollment: *PendingCsr*

Table 244: DELETE CSR Generation Pending Input Parameters

Name	In	Description
ids	Body	**Required**. An array of 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 on the next page) to retrieve a list of all the pending CSRs to determine the CSR IDs.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.12.4 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 Overview) are required to use this feature: CertificateEnrollment: *PendingCsr*

Table 245: 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 246: 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 containing the subject of the certificate including the certificate subject information, the subject alternative names, the key length, and the hash algorithm.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.12.5 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 610), 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 271).

Tip: The following permissions (see Security Overview) are required to use this feature:
CertificateEnrollment: *CsrGeneration*

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 247: POST CSR Generation Generate Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></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-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 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 address of the certificate to be requested with the CSR.</td></tr>
</table></td></tr>
<tr><td>KeyType</td><td>Body</td><td>Required. A string indicating the desired key encryption of the certificate. Accepted key types are:
• RSA</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>• ECC</td></tr>
<tr><td>KeyLength</td><td>Body</td><td>Required. An integer indicating the desired key size of the certificate. Accepted key sizes are:
• 256
• 384
• 521
• 2048
• 4096
• 8192</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.

  Tip: Although you can include a template in your CSR, template handling in CSRs is future functionality, and the template will not be parsed back out of the CSR. Instead, submit a template directly with your CSR enrollment (see POST Enrollment CSR on page 610).</td></tr>
<tr><td>SANs</td><td>Body</td><td>An array of key/value pairs that represent the elements for Keyfactor Command to use when generating the subject alternative name (SAN) for the certificate requested by the CSR. Possible values for the key are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>rfc822</td><td>RFC 822 Name</td></tr><tr><td>dns</td><td>DNS Name</td></tr><tr><td>directory</td><td>Directory Name</td></tr><tr><td>uri</td><td>Uniform Resource Identifier</td></tr><tr><td>ip4</td><td>IP v4 Address</td></tr><tr><td>ip6</td><td>IP v6 Address</td></tr><tr><td>registeredid</td><td>Registered ID (an OID)</td></tr><tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr><tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr></table>
For example:</td></tr>
</table>

Name	In	Description
		<pre>"SANs": { "dns": ["dnssan1.keyexample.com", "dnssan2.keyexample.com", "dnssan3.keyexample.com"], "ip4": ["192.168.2.73"] }</pre>

Table 248: POST CSR Generation Generate Response Data

Name	Description
CSR	The text of the CSR in PEM format.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.13 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 249: Custom Job Types Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the custom job type for the specified ID.	DELETE Custom Job Types ID on the next page
/{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 565
/	POST	Creates a custom job type.	POST Custom Job Types on page 567

Endpoint	Method	Description	Link
/	PUT	Updates an existing custom job type.	PUT Custom Job Types on page 571

2.2.13.1 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 Overview) are required to use this feature:
AgentManagement: *Modify*

Table 250: 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 the next page) to retrieve a list of all the custom job types to determine the job type GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.13.2 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 Overview) are required to use this feature:
AgentManagement: *Read*

Table 251: 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 the next page) to retrieve a list of all the custom job types to determine the job type GUID.

Table 252: GET JobTypes Custom {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>JobTypeName</td><td>A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it.</td></tr>
<tr><td>Description</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>A value that indicates the data type of the job type field.
Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table></td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false).</td></tr>
</table></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.13.3 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 Overview) are required to use this feature:
AgentManagement: *Read*

Table 253: GET JobTypes 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 254: GET JobTypes Custom Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>JobTypeName</td><td>A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it.</td></tr>
<tr><td>Description</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>A value that indicates the data type of the job type field.
Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table></td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false).</td></tr>
</table></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.13.4 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 Overview) are required to use this feature:
AgentManagement: *Modify*

Table 255: 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:
<pre>
"JobTypeFields": [
   {
      "Name": "Favorite Type of Pet",
</pre></td></tr>
</table>

Name	In	Description
		"Type": "String", "DefaultValue": "Cat", "Required": true }, { "Name": "Model Year of First Car", "Type": "Int" }, { "Name": "Mother's Birthday", "Type": "DateTime" }]

Table 256: POST JobTypes Custom Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>JobTypeName</td><td>A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it.</td></tr>
<tr><td>Description</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>A value that indicates the data type of the job type field.
Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table></td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false).</td></tr>
</table></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.13.5 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 Overview) are required to use this feature:
AgentManagement: *Modify*

Warning: 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 257: PUT JobTypes Custom Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Id</td><td>Body</td><td>The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>JobTypeName</td><td>Body</td><td>Required. A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it. This name should not contain spaces.</td></tr>
<tr><td>Description</td><td>Body</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>Body</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>Required. A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>Required. A value that indicates the data type of the job type field.
It may be entered as either an integer or the matching enum value. Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table></td></tr>
<tr><td>DefaultValue</td><td>Required*. A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.
This field is required if the Required parameter is set to true.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false). The default is false.</td></tr>
</table>
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 258: PUT JobTypes Custom Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID for the custom job type. This ID is automatically set by Keyfactor Command.</td></tr>
<tr><td>JobTypeName</td><td>A string containing the short name for the custom job type. This is used to reference the job type when submitting a job for it.</td></tr>
<tr><td>Description</td><td>A string containing a description for the custom job type.</td></tr>
<tr><td>JobTypeFields</td><td>An array of job type fields that indicate the type of tasks the job type is designed to perform. Job type fields parameters are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string that indicates the name for the job type field.</td></tr>
<tr><td>Type</td><td>A value that indicates the data type of the job type field.
Possible values are:
<table>
<tr><th>Integer Value</th><th>Enum Value</th><th>Description</th></tr>
<tr><td>1</td><td>String</td><td>String</td></tr>
<tr><td>2</td><td>Int</td><td>Integer</td></tr>
<tr><td>3</td><td>DateTime</td><td>Date</td></tr>
<tr><td>4</td><td>Bool</td><td>Boolean</td></tr>
</table></td></tr>
<tr><td>DefaultValue</td><td>A string containing the default value of the job type field.
If Type is Boolean, this field should contain true or false.</td></tr>
<tr><td>Required</td><td>A Boolean that sets whether the job type field is required (true) or not (false).</td></tr>
</table></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14 Enrollment

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

Table 259: Enrollment Endpoints

Endpoint	Method	Description	Link
/Settings/{Id}	GET	Returns the template settings to use during enrollment.	GET Enrollment Settings ID below
/CSR/Context/My	GET	Returns the templates available for CSR enrollment by the current user.	GET Enrollment CSR Content My on page 583
/PFX/Context/My	GET	Returns the templates available for PFX enrollment by the current user.	GET Enrollment PFX Content My on page 595
/AvailableRenewal/Id/{id}	GET	Returns the type of renewals available for the referenced certificate ID.	GET Enrollment Available Renewal ID on page 607
/AvailableRenewal/Thumbprint/{thumbprint}	GET	Returns the type of renewals available for the referenced certificate thumbprint.	GET Enrollment Available Renewal Thumbprint on page 608
/CSR	POST	Performs a CSR enrollment.	POST Enrollment CSR on page 610
/PFX	POST	Performs a PFX enrollment.	POST Enrollment PFX on page 616
/CSR/Parse	POST	Returns information found in a CSR in a human friendly form.	POST Enrollment CSR Parse on page 629
/PFX/Deploy	POST	Adds a certificate into a certificate store following a PFX enrollment or certificate renewal.	POST Enrollment PFX Deploy on page 631
/PFX/Replace	POST	Replaces a certificate in a certificate store following a PFX enrollment.	POST Enrollment PFX Replace on page 636
/Renew	POST	Performs a certificate renewal.	POST Enrollment Renew on page 639

2.2.14.1 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 Overview) are required to use this feature: CertificateEnrollment: *EnrollCSR* or CertificateEnrollment: *EnrollPFX* or CertificateEnrollment: *CsrGeneration*

Table 260: GET Enrollment Settings {id} Input Parameters

Name	Description
id	The enrollment template Id. Use the *GET /Templates* method (see GET Templates on page 1206) to retrieve a list of all the templates to determine the template ID.

Table 261: GET Enrollment Settings {id} Response Body

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateRegexes</td><td>An object containing the regular expressions resolved for the template. Regular expression details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see <a>GET Templates Subject Parts on page 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the organization name entered in the field be one of "Key Example Inc", "Key Example" or "Key Example Inc.":
<code>^(?:Key Example Inc|Key Example|Key Example, Inc\.)$</code>
The period in the final company name (Key Example, Inc.) needs to be escaped in the regular expression</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Subject Part	Example
	with a slash ("\") but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.`

<table>
<tr><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>(?:keyexample1\.com|keyexample2\.com)$</pre></td></tr>
<tr><td>IPv4 (Subject Alternative Name: IPv4 Address)</td><td>This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers:<pre>^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$</pre>This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:<pre>^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$</pre></td></tr>
<tr><td>IPv6 (Subject Alternative Name: IPv6 Address)</td><td>This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons:<pre>^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$</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 lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com":<pre>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</pre></td></tr>
<tr><td>UPN (Subject Alternative Name: User Principal Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com":<pre>^[a-zA-Z0-9'_\.\-]*@keyexample\.com$</pre></td></tr>
</table>
</td></tr>
<tr><td>Error</td><td>A string specifying the error message displayed to the user when the subject</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>part referenced in the CSR or entered for a PFX enrollment does not match the given regular expression. Note that 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 object 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 GET Templates Settings on page 1186. 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 1205) 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": [
   {
      "SubjectPart": "L",
      "Value": "Denver"
   },
   {
      "SubjectPart": "ST",
      "Value": "Colorado"
   }
]
</pre>
</td></tr>
<tr><td>TemplatePolicy</td><td>An array containing the template policy settings. The template policy array contains the following parameters:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>RSAValidKeySizes</td><td>An object containing a comma-delimited list of integers defining the valid RSA key sizes supported for all templates used for enrollment. The supported values are:</td></tr>
</table>
</td></tr>
</table>

Name	Description

Value	Description
	• 2048 • 4096
ECCValidCurves	An object containing a list of strings defining the valid elliptic curve algorithms for ECC templates. These 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").
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 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.

For example:

```
"TemplatePolicy": {
   "RSAValidKeySizes": [
      2048,
      4096
   ],
   "ECCValidCurves": [
      "1.2.840.10045.3.1.7",
      "1.3.132.0.34"
      "1.3.132.0.35"
   ],
   "AllowKeyReuse": false,
   "AllowWildcards": true,
   "RFCEnforcement": true,
```

Name	Description
	<pre>"AllowEd448": false, "AllowEd25519": false }</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.2 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. 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 Overview) are required to use this feature:
CertificateEnrollment: *EnrollCSR*

Table 262: GET Enrollment CSR Content My Response Body

Name	Description
Templates	An array containing the templates available for enrollment by the user. The array 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.
Forest	A string containing the name of the configuration tenant the template is associated with.
KeySize	A string indicating the minimum supported key size of the template.
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 certificate authorities from which the template is available for enrollment, that are configured for enrollment in Keyfactor Command, and on which the requesting user has enrollment permissions. Information about the CA includes:

Name	Description
Name	The full name of the CA, made up of the DNS hostname of the certificate authority (e.g.

<table>
<tr><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>corpca01.keyexample.com) and the logical name (e.g. CorpIssuingCA1) for a full name similar to corpca01.keyexample.com\\CorpIssuingCA1.</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>
</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</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**
	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: **Value** / **Description** 1 — String: A free-form data entry field. 2 — Multiple Choice: Provides a list of acceptable values for the field. The multiple choice values are provided in the *Options* parameter. For example: `"EnrollmentFields": [ { "Id": 3, "Name": "MyCustomField", "Options": ["Green","Red","Yellow","Blue"], "DataType": 2 }`

<table>
<tr><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>]</pre></td></tr>
<tr><td>MetadataFields</td><td>An object 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 field settings array contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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</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>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>
<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>
<table>
<tr><th>Name</th><th>Description</th></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>
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 object 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>Template-</td><td>The Keyfactor Command reference ID of the certificate template</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>Id</td><td>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 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_
\.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name

Description

Name

Description

Subject Part	Example
	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: `^(?:US|CA)$`

Name	Description
	Name / **Description**

Name

Description

Name

Description

Subject Part	Example
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$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
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

Name	Description
	Name / **Description**

Name

Description

Name

Description

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

<table>
<tr><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>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 that 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>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 containing enrollment information for standalone certificate authorities available for enrollment for the current user. Information about the CA includes:
<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>
</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.3 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. 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 Overview) are required to use this feature:
CertificateEnrollment: *EnrollPFX*

Table 263: GET Enrollment PFX Content My Response Body

Name	Description
Templates	An array containing the templates available for enrollment by the user. The array 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.
Forest	A string containing the name of the configuration tenant the template is associated with.
KeySize	A string indicating the minimum supported key size of the template.
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 certificate authorities from which the template is available for enrollment, that are configured for enrollment in Keyfactor Command, and on which the requesting user has enrollment permissions. Information about the CA includes:

Name	Description
Name	The full name of the CA, made up of the DNS hostname of the certificate authority (e.g.

<table>
<tr><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>corpca01.keyexample.com) and the logical name (e.g. CorpIssuingCA1) for a full name similar to corpca01.keyexample.com\\CorpIssuingCA1.</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>
</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</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>
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:

<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": [
   {
      "Id": 3,
      "Name": "MyCustomField",
      "Options": ["Green","Red","Yellow","Blue"],
      "DataType": 2
   }
</pre>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>]</pre></td></tr>
<tr><td>MetadataFields</td><td>An object 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 field settings array contains the following parameters:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>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</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>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>
<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>
<table>
<tr><th>Name</th><th>Description</th></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>
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 object 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>Template-</td><td>The Keyfactor Command reference ID of the certificate template</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>Id</td><td>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 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_
\.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name

Description

Name

Description

Subject Part	Example
	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: `^(?:US|CA)$`

Name	Description
	Name / **Description**

Name

Description

Name

Description

Subject Part	Example
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$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
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

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

<table>
<tr><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>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 that 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>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 containing enrollment information for standalone certificate authorities available for enrollment for the current user. Information about the CA includes:
<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>
</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.4 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 Overview) are required to use this feature:
Certificates: *Read*

At either the global or collection level. See note under CollectionId, below.

Table 264: 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 265: GET Enrollment Available Renewal ID {id} Response Body

<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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.5 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 Overview) are required to use this feature:
Certificates: *Read*

At either the global or collection level. See note under CollectionId, below.

Table 266: 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 267: GET Enrollment Available Renewal Thumbprint {thumbprint} Response Body

<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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.6 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 Overview) are required to use this feature:
CertificateEnrollment: *EnrollCSR*

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).

Table 268: 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. 2021-05-19T16:23:01Z).
Template	Body	**Required***. A string that sets the name of the certificate template that should be used to issue the certificate. The template short name should be used. This field is **required** unless the enrollment is being done against a standalone CA.
CertificateAuthority	Body	**Required***. A string that sets the name of the certificate authority that will be used to enroll against if there is more than one available with the provided template name. The certificate authority name can either be provided in *hostname**logical name* format or as just the *logical name*. For example: corpca01.keyexample.com\\CorpIssuingCA1 **OR** CorpIssuingCA1 If no certificate authority is provided, one will be chosen at random from the certificate authorities available for enrollment with the provided *Template*. This field is optional unless the enrollment is being done against a standalone CA, in which case it is **required**.
IncludeChain	Body	A Boolean that sets whether to include the certificate chain in the response (true) or not (false). The default is *false*.
Metadata	Body	An array of key/value pairs 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": {`

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

```
   "AppOwnerFirstName": "William",  // This is a String
field.
   "AppOwnerLastName": "Smith",
   "AppOwnerEmailAddress": "willi-
am.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.
</td></tr>
<tr><td>SANs</td><td>Body</td><td>An array of key/value pairs that represent the elements for Keyfactor Command to use when generating the subject alternative name (SAN) for the certificate requested by the CSR. Possible values for the key are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>rfc822</td><td>RFC 822 Name</td></tr>
<tr><td>dns</td><td>DNS Name</td></tr>
<tr><td>directory</td><td>Directory Name</td></tr>
<tr><td>uri</td><td>Uniform Resource Identifier</td></tr>
<tr><td>ip4</td><td>IP v4 Address</td></tr>
<tr><td>ip6</td><td>IP v6 Address</td></tr>
<tr><td>registeredid</td><td>Registered ID (an OID)</td></tr>
<tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr>
<tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr>
</table>
For example:</td></tr>
</table>

Name	In	Description
		<pre>"SANs": { "dns": ["dnssan1.keyexample.com", "dnssan2.keyexample.com", "dnssan3.keyexample.com"], "ip4": ["192.168.2.73"] }</pre> **Note:** Entering SANs with this option may either append or overwrite the SANs in the CSR request depending on how the issuing CA is configured. Please be sure to check that the certificate has the correct SANs after issuance. Any SAN added automatically as a result of the RFC 2818 compliance settings (see GET Templates on page 1206) 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.
AdditionalEnrollmentFields	Body	An array of key/value pairs 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": "ValueOne", "CustomMultiChoiceTwo": "ValueTwo" }` See *Configuring Template Options* of the *Keyfactor Command Reference Guide* for more information.
x-CertificateFormat	Header	**Required**. The desired output format for the certificate. Available options are DER and PEM.

Table 269: POST Enrollment CSR Response Data

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateInformation</td><td>Information about the certificate that was requested. CSR information includes:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SerialNumber</td><td>A string indicating the serial number of the certificate.</td></tr>
<tr><td>IssuerDN</td><td>A string indicating the issuer DN of the certificate.</td></tr>
<tr><td>Thumbprint</td><td>A string indicating the thumbprint of the certificate.</td></tr>
<tr><td>KeyfactorID</td><td>An integer indicating the Keyfactor Command reference ID of the issued certificate.</td></tr>
<tr><td>Certificates</td><td>An array of certificates in the order of:
end entity
intermediate CA
Root CA
Intermediate CA and root CA certificates will only be included if the request parameter IncludeChain was set to true.</td></tr>
<tr><td>WorkflowInstanceId</td><td>A string containing the Keyfactor Command reference GUID of the workflow instance.
Tip: Both the WorkflowInstanceId and the WorkflowReferenceId refer to the same workflow instance record—one using a GUID and one using a more human readable integer.</td></tr>
<tr><td>WorkflowReferenceId</td><td>An integer containing the Keyfactor Command reference ID of the workflow instance.
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>
</table></td></tr>
</table>

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.7 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 Overview) are required to use this feature:
CertificateEnrollment: *EnrollPFX*

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 627 parameter) using the POST /Enrollment/PFX/Deploy method (see POST Enrollment PFX Deploy on page 631) or POST /Enrollment/PFX/Replace method (see POST Enrollment PFX Replace on page 636).

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).

This method has two available versions. Keyfactor recommends using the newer method when possible. For more information about versioning, see Versioning on page 6.

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 *PopulateMissingValuesFromAD* parameter has been removed from the version 2 endpoint.

Table 270: POST Enrollment PFX v2 Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Stores</td><td>Body</td><td>An object containing a comma delimited set of arrays 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>An array of GUIDs indicating the certificate store(s) to which the certificate should be deployed.
Use the GET /CertificateStores method (see <a>GET Certificate Stores on page 384) 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 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 <a>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 224) 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 key/value pairs 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:
<code>"JobProperties": ["NetscalerVserver"]</code></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>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>
</td></tr>
<tr><td>CustomFriendlyName</td><td>Body</td><td>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.</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>Subject</td><td>Body</td><td>Required*. A string containing the subject name using X.500 format. For example:
<code>"Subject": "CN=websrvr14.keyexample.com,OU=IT,O=\"Key Example, Inc.\",L=Independence,ST=OH,C=US"</code>
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>IncludeChain</td><td>Body</td><td>A Boolean that sets whether to include the certificate chain in the response (true) or not (false). The default is true.</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</td></tr>
</table>

Name	In	Description
		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.
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**.
Metadata	Body	An array of key/value pairs 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.
Timestamp	Body	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. 2021-05-19T16:23:01Z).
Template	Body	**Required***. A string that sets the name of the certificate template that

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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 array of key/value pairs that represent the elements for Keyfactor Command to use when generating the subject alternative name (SAN) for the certificate requested by the CSR. 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>InstallIn-toExistingCertificateStores</td><td>Body</td><td>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 InstallIn-toExistingCertificateStores parameter to make the determination as to</td></tr>
</table>

Name	In	Description
		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.
AdditionalEnrollmentFields	Body	An array of key/value pairs 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.
x-CertificateFormat	Header	**Required**. The desired output format for the certificate. Available options are PFX, Zip, and Store. If Store is selected, no certificate blob will be returned in the response. The Store option is designed to be used when pushing a newly obtained PFX certificate to a certificate store (see POST Enrollment PFX Deploy on page 631).

Table 271: POST Enrollment PFX v2 Response Data

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SuccessfulStores</td><td>An object containing a comma delimited list of certificate stores, referenced by certificate store GUID, to which the certificate was successfully scheduled for deployment.</td></tr>
<tr><td>CertificateInformation</td><td>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 used when pushing a newly obtained PFX certificate to a certificate store (see POST Enrollment PFX Deploy on page 631).</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</td></tr>
</table></td></tr>
</table>

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

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 272: POST Enrollment PFX v1 Input Parameters

Name	In	Description
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.
Password	Body	**Required**. A string that sets the password used to encrypt the contents of the PFX file. The minimum password length is controlled by the *Password Length* application setting. The default is *12*. See Application Settings: Enrollment Tab 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*.
Subject	Body	**Required***. A string containing the subject name using X.500 format. For example: `"Subject": "CN=webservr14.keyexample.com,OU=IT,O=\"Key Example, Inc.\",L=Independence,ST=OH,C=US"` This field is **required** if the *Common Name Regular Expression* application setting is set to the default value of *.+*. See Application Settings: Enrollment Tab in the *Keyfactor Command Reference Guide* for more information.
IncludeChain	Body	A Boolean that sets whether to include the certificate chain in the response (true) or not (false). The default is *true*.
RenewalCertificateId	Body	An integer that sets the ID of the certificate to be renewed when the method is called on a certificate renewal.
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*.

Name	In	Description
		This field is optional unless the enrollment is being done against a standalone CA, in which case it is **required**.
Timestamp	Body	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. 2021-05-19T16:23:01Z).
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.
Metadata	Body	An array of key/value pairs 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.
SANs	Body	An array of key/value pairs that represent the elements for Keyfactor Command to use when generating the subject alternative name (SAN) for the certificate requested by the CSR. Possible values for the key are:

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>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>AdditionalEnrollmentFields</td><td>Body</td><td>An array of key/value pairs 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.</td></tr>
<tr><td>x-CertificateFormat</td><td>Header</td><td>Required. The desired output format for the certificate. Available options are PFX, Zip, and Store. If Store is selected, no certificate blob will be returned in the response. The Store option is designed to be used when pushing a newly obtained PFX certificate to a certificate store (see POST Enrollment PFX Deploy on page 631).</td></tr>
</table>

Table 273: POST Enrollment PFX v1 Response Data

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>CertificateInformation</td><td>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 used when pushing a newly obtained PFX certificate to a certificate store (see POST Enrollment PFX Deploy on page 631).</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>
</table></td></tr>
</table>

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.8 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.

Table 274: 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 275: POST Enrollment CSR Parse Response Data

Name	Description
(CSR Contents)	An array containing key/value pairs representing 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 Command (e.g. URL) and will only be found in CSRs generated outside Keyfactor Command.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.9 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 616) 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 Overview) are required to use this feature:
CertificateStoreManagement: *Schedule*
CertificateEnrollment: *EnrollPFX*

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 276: 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 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>An array of GUIDs indicating the certificate store(s) to which the certificate should be deployed.
Use the GET /CertificateStores method (see <a>GET Certificate Stores on page 384) 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>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 224) 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 key/value pairs 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:
<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.</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>The setting is referenced using the following format:
<code>"Properties": {"Nets-calerVserver":"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.
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>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. 2021-05-19T16:23:01Z). 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 the certificate store GUIDs for the stores to which the certificate should be added.</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>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 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>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>
<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>
</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>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>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 224) 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 key/value pairs 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:
<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</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		Name / **Description** 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.

Table 277: POST Enrollment PFX Deploy Response Data

Name	Description
SuccessfulStores	An array of 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 GUIDs for the certificates stores for which management jobs to deploy the certificate could not be created.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.10 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 *Store* value for the *x-certificateformat* header (see POST Enrollment PFX on page 616) 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 Overview) are required to use this feature:
CertificateStoreManagement: *Schedule*
CertificateEnrollment: *EnrollPFX*

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 631).

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 278: 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	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. 2021-05-19T16:23:01Z). 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 279: POST Enrollment PFX Replace Response Data

Name	Description
SuccessfulStores	An array of 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 GUIDs for the certificates stores for which management jobs to deploy the certificate could not be created.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.14.11 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 Overview) are required to use this feature:
Certificates: *Read*
CertificateEnrollment: *EnrollPFX*

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 631) or POST /Enrollment/PFX/Replace method (see POST Enrollment PFX Replace on page 636).

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).

Table 280: 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. 2021-05-19T16:23:01Z).
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 *host-name**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 607 or GET Enrollment Available Renewal Thumbprint on page 608).
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 607 or GET Enrollment Available Renewal Thumbprint on page 608).

Table 281: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.15 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 282: License Endpoint

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

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

Tip: The following permissions (see Security Overview) are required to use this feature:
SystemSettings: *Read*

Table 283: 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 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 version of the license. MinorRev — A string indicating the valid incremental release version of the license.

Name	Description
	Name / **Description** LicensedFeatures — An array containing the Keyfactor Command features included in the license. **Name** / **Description** FeatureID — A string indicating the ID code of feature. DisplayName — A string indicating the name of the feature as displayed on the license page in the Management Portal. Enabled — A Boolean that indicates whether the feature is enabled (true) or not (false). Quantity — 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. ExpirationDate — This field is unused and will always return *null*.

Tip: Currently there is only one licensed product offered, which is Keyfactor Command.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.16 MacEnrollment

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

Table 284: 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

2.2.16.1 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.

Tip: The following permissions (see Security Overview) are required to use this feature:
SystemSettings: *Read*

Table 285: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.16.2 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 Overview) are required to use this feature:
SystemSettings: *Modify*

Warning: 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 286: PUT MacEnrollment Response Data

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 in the *Keyfactor Command Reference Guide* 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 287: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.17 MetadataFields

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

Table 288: MetadataFields Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes an existing metadata field.	DELETE MetadataFields ID on the next page
/{id}	GET	Returns detailed information for the specified metadata field.	GET MetadataFields ID on page 649
/{name}	GET	Returns detailed information for the specified metadata field.	GET MetadataFields Name on page 652

Endpoint	Method	Description	Link
/{id}/InUse	GET	Returns a Boolean stating whether the metadata type is associated with a certificate.	GET MetadataFields ID InUse on page 655
/	DELETE	Deletes multiple metadata fields specified in the request body.	DELETE MetadataFields on page 656
/	GET	Returns all metadata field types with paging (number of pages to return and number of results per page) options.	GET MetadataFields on page 656
/	POST	Creates a new metadata field using values supplied in the request body.	POST MetadataFields on page 660
/	PUT	Updates an existing metadata field using values supplied in the request body.	PUT MetadataFields on page 666

2.2.17.1 DELETE MetadataFields 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 Overview) are required to use this feature:
CertificateMetadataTypes: *Modify*

Table 289: DELETE MetadataFields {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 MetadataFields on page 656) 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 MetadataFields ID InUse on page 655) to determine whether a metadata field is in use.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.17.2 GET MetadataFields 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 Overview) are required to use this feature: CertificateMetadataTypes: *Read*

Table 290: GET MetadataFields {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 MetadataFields on page 656) to retrieve a list of all the metadata fields to determine the metadata field's ID.

Table 291: GET MetadataFields {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>
<tr><td>7</td><td>Email</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".
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</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 1173).</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 1173).</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 1173).</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.

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 1173).</td></tr>
<tr><td>DefaultValue</td><td>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</td></tr>
</table>

Name	Description
	email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, *Email* 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 1173).
AllowAPI	A Boolean that sets whether methods in the Classic API can be used to manipulate data in the metadata record (true) or not (false). The default is *true*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
ExplicitUpdate	A Boolean that sets whether methods in the Classic API must submit an overwrite flag in the request in order to overwrite an existing value in the metadata record (true) or not (false). The default is *false*. This setting does not apply to the Keyfactor API. 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.17.3 GET MetadataFields 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 Overview) are required to use this feature:
CertificateMetadataTypes: *Read*

Table 292: GET MetadataFields {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 293: GET MetadataFields {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>
<tr><td>7</td><td>Email</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".
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</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 1173).</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 1173).</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 1173).</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.

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 1173).</td></tr>
<tr><td>DefaultValue</td><td>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</td></tr>
</table>

Name	Description
	email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, *Email* 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 1173).
AllowAPI	A Boolean that sets whether methods in the Classic API can be used to manipulate data in the metadata record (true) or not (false). The default is *true*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
ExplicitUpdate	A Boolean that sets whether methods in the Classic API must submit an overwrite flag in the request in order to overwrite an existing value in the metadata record (true) or not (false). The default is *false*. This setting does not apply to the Keyfactor API. 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.17.4 GET MetadataFields ID InUse

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 Overview) are required to use this feature:
CertificateMetadataTypes: *Read*

Table 294: GET MetadataFields {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 MetadataFields on the next page) to retrieve a list of all the metadata fields to determine the metadata field's ID.

Table 295: GET MetadataFields {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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.17.5 DELETE MetadataFields

The DELETE /MetadataFields method is used to delete multiple metadata fields in one request. The metadata fields IDs should be supplied in the request body as a JSON array of integers. 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 Overview) are required to use this feature:
CertificateMetadataTypes: *Modify*

Table 296: DELETE MetadataFields Input Parameters

Name	In	Description
ids	Body	**Required**. An array of Keyfactor Command reference IDs for the metadata fields to be deleted. Use the *GET /MetadataFields* method (see GET MetadataFields 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 MetadataFields ID InUse on the previous page) to determine whether a metadata field is in use.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.17.6 GET MetadataFields

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 Overview) are required to use this feature:
CertificateMetadataTypes: *Read*

Table 297: GET MetadataFields 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Logons Search*. 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 *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 298: GET MetadataFields 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>
<tr><td>7</td><td>Email</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".
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</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 1173).</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 1173).</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 1173).</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.

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 1173).</td></tr>
<tr><td>DefaultValue</td><td>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</td></tr>
</table>

Name	Description
	email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, *Email* 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 1173).
AllowAPI	A Boolean that sets whether methods in the Classic API can be used to manipulate data in the metadata record (true) or not (false). The default is *true*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
ExplicitUpdate	A Boolean that sets whether methods in the Classic API must submit an overwrite flag in the request in order to overwrite an existing value in the metadata record (true) or not (false). The default is *false*. This setting does not apply to the Keyfactor API. 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.17.7 POST MetadataFields

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 Overview) are required to use this feature:
CertificateMetadataTypes: *Modify*

Table 299: POST MetadataFields 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>
<tr><td>7</td><td>Email</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, 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>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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 1173).</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 1173).</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).
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 1173).</td></tr>
<tr><td>Options</td><td>Body</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 required for metadata fields with data type multiple choice. For other data types, it will be ignored.</td></tr>
</table>

Name	In	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 1173).
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*, *Email* 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 1173).
AllowAPI	Body	A Boolean that sets whether methods in the Classic API can be used to manipulate data in the metadata record (true) or not (false). The default is *true*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
ExplicitUpdate	Body	A Boolean that sets whether methods in the Classic API must submit an overwrite flag in the request in order to overwrite an existing value in the metadata record (true) or not (false). The default is *false*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
DisplayOrder	Body	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 300: POST MetadataFields 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>
<tr><td>7</td><td>Email</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".
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</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 1173).</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 1173).</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 1173).</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.

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 1173).</td></tr>
<tr><td>DefaultValue</td><td>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</td></tr>
</table>

Name	Description
	email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, *Email* 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 1173).
AllowAPI	A Boolean that sets whether methods in the Classic API can be used to manipulate data in the metadata record (true) or not (false). The default is *true*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
ExplicitUpdate	A Boolean that sets whether methods in the Classic API must submit an overwrite flag in the request in order to overwrite an existing value in the metadata record (true) or not (false). The default is *false*. This setting does not apply to the Keyfactor API. 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.17.8 PUT MetadataFields

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 Overview) are required to use this feature:
CertificateMetadataTypes: *Modify*

Warning: 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 301: PUT MetadataFields 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>
<tr><td>7</td><td>Email</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, underscores, periods, and/or hyphens followed by exactly either "@keyexample.org" or "keyexample.com".</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><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 1173).</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 1173).</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).

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 1173).</td></tr>
<tr><td>Options</td><td>Body</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 required for metadata fields with data type multiple choice. For other data</td></tr>
</table>

Name	In	Description
		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 1173).
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*, *Email* 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 1173).
AllowAPI	Body	A Boolean that sets whether methods in the Classic API can be used to manipulate data in the metadata record (true) or not (false). The default is *true*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
ExplicitUpdate	Body	A Boolean that sets whether methods in the Classic API must submit an overwrite flag in the request in order to overwrite an existing value in the metadata record (true) or not (false). The default is *false*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
DisplayOrder	Body	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 302: PUT MetadataFields 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>
<tr><td>7</td><td>Email</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".
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</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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 1173).</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 1173).</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 1173).</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.

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 1173).</td></tr>
<tr><td>DefaultValue</td><td>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</td></tr>
</table>

Name	Description
	email addresses (limit 100 characters per email address). This field is only supported for metadata fields with data types *string*, *integer*, *Boolean*, *Email* 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 1173).
AllowAPI	A Boolean that sets whether methods in the Classic API can be used to manipulate data in the metadata record (true) or not (false). The default is *true*. This setting does not apply to the Keyfactor API. This is considered deprecated and may be removed in a future release.
ExplicitUpdate	A Boolean that sets whether methods in the Classic API must submit an overwrite flag in the request in order to overwrite an existing value in the metadata record (true) or not (false). The default is *false*. This setting does not apply to the Keyfactor API. 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.18 Monitoring Revocation

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

Table 303: Monitoring Revocation Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the revocation monitoring location with the specified ID.	DELETE Monitoring Revocation ID on the next page
/{id}	GET	Returns details for the revocation monitoring location with the specified ID.	GET Monitoring Revocation ID on page 674
/	PUT	Edits the revocation monitoring location with the specified ID.	PUT Monitoring Revocation on page 687

Endpoint	Method	Description	Link
/	GET	Returns details for all revocation monitoring location according to the provided filter and output parameters.	GET Monitoring Revocation on page 677
/	POST	Creates a new revocation monitoring location.	POST Monitoring Revocation on page 681
/ResolveOSCP	POST	Resolves the given OCSP certificate authority.	POST Monitoring Resolve OSCP on page 693
/Test	POST	Tests the revocation monitoring alert with the specified ID.	POST Monitoring Revocation Test on page 694
/TestAll	POST	Tests the revocation monitoring alerts.	POST Monitoring Revocation Test All on page 696

2.2.18.1 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 304: 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 677) to retrieve a list of all the revocation monitoring locations to determine the ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.18.2 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 305: 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 677) to retrieve a list of all the revocation monitoring locations to determine the ID.

Table 306: 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>For CRL endpoints only, an array 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 object containing a list of strings with email addresses to which the email reminders should be sent.</td></tr></table></td></tr>
<tr><td>Dashboard</td><td>An array 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.
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 array containing the inventory schedule set for the revocation monitoring location. Supported</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>

Note: Although the 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 array indicating the OCSP endpoint configuration. OCSP endpoint details are:</td></tr>
</table>

Name	Description
	<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. Use the *GET /CertificateAuthority* method (see GET Certificate Authority on page 292) to retrieve a list of all the CAs to determine the ID. This value will be null on a response if the endpoint was configured using the *CertificateContents* 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 base 64 encoded SHA1 hash of the *AuthorityName*.</td></tr><tr><td>AuthorityKeyId</td><td>A 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 CA.</td></tr></table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.18.3 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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 307: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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 308: GET Monitoring Revocation Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the revocation monitoring location.
Name	A string indicating the name of the revocation monitoring location.
EndpointType	A string indicating the type of revocation monitoring endpoint: OCSP or CRL.
Location	A string indicating the location for the revocation monitoring endpoint. For CRL endpoints, this can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location. For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.
Email	For CRL endpoints only, an array indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are: **Value** / **Description** EnableReminder — A Boolean indicating whether to send email reminders for this location (true) or not (false). WarningDays — An integer indicating the number of days before expiration to send the warning email. Recipients — An object containing a list of strings with email addresses to which the email reminders should be sent.
Dashboard	An array indicating the configuration for display on the dashboard. Dashboard details are: **Value** / **Description** Show — A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false). WarningHours — An integer indicating the number of hours prior to expiration when the location begins to appear in a warning state on the dashboard. *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.
Schedule	An array containing the inventory schedule set for the revocation monitoring location. Supported

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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 array indicating the OCSP endpoint configuration. OCSP endpoint details are:</td></tr>
</table>

Name	Description
	<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. Use the *GET /CertificateAuthority* method (see GET Certificate Authority on page 292) to retrieve a list of all the CAs to determine the ID. This value will be null on a response if the endpoint was configured using the *CertificateContents* 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 base 64 encoded SHA1 hash of the *AuthorityName*.</td></tr><tr><td>AuthorityKeyId</td><td>A 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 CA.</td></tr></table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.18.4 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 309: POST Monitoring Revocation Input Parameters

Name	In	Description
Name	Body	**Required**. A string indicating the name of the revocation monitoring location.
EndpointType	Body	**Required**. A string indicating the type of revocation monitoring endpoint: OCSP or CRL.
Location	Body	**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. For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.
Email	Body	**Required*.** for CRL endpoints. For CRL endpoints only, an array indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are: **Value** / **Description** EnableReminder — A Boolean indicating whether to send email reminders for this location (true) or not (false). The default is false. WarningDays — An integer indicating the number of days before expiration to send the warning email. Recipients — An object containing a list of strings with email addresses to which the email reminders should be sent.
Dashboard	Body	**Required**. An array indicating the configuration for display on the dashboard. Dashboard details are: **Value** / **Description** Show — **Required**. A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false). The default is false. WarningHours — **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.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Schedule</td><td>Body</td><td>An array 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 time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>OCSPPara-meters</td><td>Body</td><td>Required*. for OCSP endpoints. For OCSP endpoints only, an array 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 indicating the certificate contents.</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 292) to retrieve a list of all the CAs to determine the ID.</td></tr>
</table>
</td></tr>
</table>

Table 310: 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>For CRL endpoints only, an array 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 object containing a list of strings with email addresses to which the email reminders should be sent.</td></tr></table></td></tr>
<tr><td>Dashboard</td><td>An array 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.
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 array containing the inventory schedule set for the revocation monitoring location. Supported schedules are:</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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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 array indicating the OCSP endpoint configuration. OCSP endpoint details are:</td></tr>
</table>

Name	Description
	<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. Use the *GET /CertificateAuthority* method (see GET Certificate Authority on page 292) to retrieve a list of all the CAs to determine the ID. This value will be null on a response if the endpoint was configured using the *CertificateContents* 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 base 64 encoded SHA1 hash of the *AuthorityName*.</td></tr><tr><td>AuthorityKeyId</td><td>A 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 CA.</td></tr></table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.18.5 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Warning: 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 311: 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.
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. For CRL endpoints only, an array 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 object containing a list of strings with 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 array 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>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</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></td><td>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 array 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 time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</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": "2022-02-25T23:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the 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 array 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 indicating the certificate contents.</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 292) to retrieve a list of all the CAs to determine the ID.</td></tr>
</table>
</td></tr>
</table>

Table 312: PUT Monitoring Revocation {id} Response Data

Name	Description
Id	An integer indicating the Keyfactor Command reference ID of the revocation monitoring location.
Name	A string indicating the name of the revocation monitoring location.
EndpointType	A string indicating the type of revocation monitoring endpoint: OCSP or CRL.
Location	A string indicating the location for the revocation monitoring endpoint. For CRL endpoints, this can be either an HTTP location or an LDAP location. Be sure to monitor the CRL locations that are in use by applications in your environment—if you're monitoring LDAP locations but applications are using an HTTP location, you're not going to receive any warning if a CRL fails to publish to the HTTP location. For OCSP endpoints, this is the full URL to the OCSP responder servicing this certificate authority's CRL.
Email	For CRL endpoints only, an array indicating the email recipients and reminder schedule for reminder alerts. Email reminder details are: **Value** / **Description** EnableReminder — A Boolean indicating whether to send email reminders for this location (true) or not (false). WarningDays — An integer indicating the number of days before expiration to send the warning email. Recipients — An object containing a list of strings with email addresses to which the email reminders should be sent.
Dashboard	An array indicating the configuration for display on the dashboard. Dashboard details are: **Value** / **Description** Show — A Boolean indicating whether to show this revocation monitoring location on the Revocation Monitoring dashboard (true) or not (false). WarningHours — An integer indicating the number of hours prior to expiration when the location begins to appear in a warning state on the dashboard. *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.
Schedule	An array containing the inventory schedule set for the revocation monitoring location. Supported

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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 array indicating the OCSP endpoint configuration. OCSP endpoint details are:</td></tr>
</table>

Name	Description
	<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. Use the *GET /CertificateAuthority* method (see GET Certificate Authority on page 292) to retrieve a list of all the CAs to determine the ID. This value will be null on a response if the endpoint was configured using the *CertificateContents* 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 base 64 encoded SHA1 hash of the *AuthorityName*.</td></tr><tr><td>AuthorityKeyId</td><td>A 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 CA.</td></tr></table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.18.6 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 Overview) are required to use this feature:
WorkflowManagement: *Modify*

Table 313: 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 292) to retrieve a list of all the CAs to determine the ID. One of either *CertificateContents* or *CertificateAuthorityId* is required, but not both.

Table 314: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.18.7 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 Overview) are required to use this feature:
WorkflowManagement: *Read*
WorkflowManagement: *Test*

Table 315: POST Monitoring Revocation Test Input Parameters

Name	In	Description
revocationMonitoringAlertTestRequest	Body	**Required**. An array containing information for the alert test. Alert test detail values are: **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, in UTC. 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*. For example: `{ "EvaluationDate": "2022-08-31T20:51:33.528Z" "SendAlerts": true }`

Table 316: POST Monitoring Revocation Test Response Data

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>RevocationMonitoringAlerts</td><td>An object containing alert details resulting from the test. Revocation monitoring alert details are:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Subject</td><td>A string indicating the email message subject for each alert. The content of this subject is not user configurable.</td></tr>
<tr><td>Message</td><td>A string indicating the email message that will be delivered for each alert. The content of this message is not user configurable.</td></tr>
<tr><td>Recipients</td><td>An object containing the recipient(s) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.18.8 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 Overview) are required to use this feature:
WorkflowManagement: *Read*
WorkflowManagement: *Test*

Table 317: POST Monitoring Revocation Test All Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>revocationMonitoringAlertTestRequest</td><td>Body</td><td>Required. An array containing information for the alert test. Alert test detail values are:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>EvaluationDate</td><td>Required. A string indicating the evaluation date/time for the test, in UTC.
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.</td></tr><tr><td>SendAlerts</td><td>A Boolean indicating whether to send alert emails with the test (true) or not (false). The default is false.</td></tr></table>For example:
<pre>{
   "EvaluationDate": "2022-08-31T20:51:33.528Z"
   "SendAlerts": true
}</pre></td></tr>
</table>

Table 318: POST Monitoring Revocation Test All Response Data

Parameter	Description
RevocationMonitoringAlerts	An object 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 object containing the recipient(s) for the alert.
AlertBuildResult	A string indicating the outcome of the test (e.g. Success).

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.19 Orchestrator Jobs

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

Table 319: 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 700
/ScheduledJobs	GET	Returns the details of active scheduled jobs, including in-process jobs.	GET Orchestrator Jobs Scheduled Jobs on page 705
/Custom	POST	Schedules a custom job on the orchestrator using the provided information.	POST Orchestrator Jobs Custom on page 709
/Reschedule	POST	Reschedules a failed orchestrator job.	POST Orchestrator Jobs Reschedule on page 713

Endpoint	Method	Description	Link
/Unschedule	POST	Unschedules an active orchestrator job.	POST Orchestrator Jobs Unschedule on page 715
/Acknowledge	POST	Sets the status of a failed orchestrator job to acknowledged.	POST Orchestrator Jobs Acknowledge on page 716
/Custom/Bulk	POST	Schedules a custom job on multiple orchestrator using the provided information.	POST Orchestrator Jobs Reschedule on page 713

2.2.19.1 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 Overview) are required to use this feature:
AgentManagement: *Read*

Tip: This method is used to return the log results from a Fetch Logs job initiated for the Keyfactor Universal Orchestrator. When used to return results for a Fetch Logs job, the last 2 MB of data from the orchestrator's log file are returned as a string in the Data field.

Tip: If jobs for the Keyfactor Universal Orchestrator fail with messages similar to the following:

```
2021-08-05 10:47:23.1940
Keyfactor.Orchestrators.JobExecutors.OrchestratorJobExecutor [Debug] - Response
status code does not indicate success: 413 (Request Entity Too Large).

at System.Net.Http.HttpResponseMessage.EnsureSuccessStatusCode() in /_
/src/System.Net.Http/src/System/Net/Http/HttpResponseMessage.cs:line 172

at Keyfactor.Orchestrators.Services.HttpService.SendPostAsync[T](String uri, Object
requestData, Dictionary`2 headers) in F:\BuildAgents\Default1\_
work\24\s\src\OrchestratorServices\HttpService.cs:line 38
```

This indicates that the amount of data being returned on the job is greater than IIS on the Keyfactor Command server is configured to accept. You will need to make modifications to the IIS settings on your Keyfactor Command server to allow it to accept larger incoming pieces of content. See *Fetch Logs* in the *Keyfactor Command Reference Guide* for more information.

Table 320: GET Orchestrator Jobs Job Status Data Input Parameters

Name	In	Description
jobHistoryId	Query	**Required**. The Keyfactor Command reference ID of the orchestrator job. Use the GET /OrchestratorJobs/JobHistory method (see GET Orchestrator Jobs Job History below) to retrieve a list of jobs to determine the job's history ID.

Table 321: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.19.2 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 Overview) are required to use this feature:
AgentManagement: *Read*

Table 322: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Job History Search Feature*. The query fields supported for this endpoint are: • *AgentId* (The GUID of the orchestrator. Run GET Agents on page 12 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 323: 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
}
```

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

<table>
<tr><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>should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce. </td></tr>
</table>
</td></tr>
<tr><td>JobType</td><td>A string indicating the job type (e.g. IISInventory).</td></tr>
<tr><td>OperationStart</td><td>The time, in UTC, at which the orchestrator job started.</td></tr>
<tr><td>OperationEnd</td><td>The time, in UTC, at which the orchestrator job finished.</td></tr>
<tr><td>Message</td><td>A string providing the error message for the operation, if any.</td></tr>
<tr><td>Result</td><td>A string indicating the result of the orchestrator job. Possible values are:
Unknown</td></tr>
</table>

Name	Description
	• 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 indicating the name of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. See *Certificate Store Operations: Adding or Modifying a Certificate Store* in the *Keyfactor Command Reference Guide* for more information.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.19.3 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 Overview) are required to use this feature:
AgentManagement: *Read*

Table 324: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Job History Search Feature*. The query fields supported for this endpoint are: • *AgentId* (The GUID of the orchestrator. Run GET Agents on page 12 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 (Capabilities in GET Agents on page 12) • *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 325: 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 indicating the name of the client machine. The value for this will vary depending on the certificate store type. For example, for a Java keystore or an F5 device, it is the hostname of the machine on which the store is located, but for an Amazon Web Services store, it is the FQDN of the Keyfactor Command Windows Orchestrator. 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. appsr-vr162.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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.19.4 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 Overview) are required to use this feature:
AgentManagement: *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 700) 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 699) to retrieve the data.

Table 326: 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 12) 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 12) 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 565) 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: **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. 2021-05-19T16:23:01Z).

<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>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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>

Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>

Note: Although the 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 array of key/value pairs 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 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.</td></tr>
</table>

Name	In	Description
		Use the *GET /JobTypes/Custom* method (see GET Custom Job Types on page 565) 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 327: 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 key/value pairs that set the values for any optional job fields configured for the custom job type.
RequestTimestmap	The date, in UTC, when the custom job was submitted.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.19.5 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 482).
- 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 709).
- 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 709).
- SSL Discovery and Monitoring
  Change the schedule on these using PUT /SSL/Networks (see PUT SSL Networks on page 1146).
- CA Synchronization for Remote CAs Managed with the Keyfactor Universal Orchestrator or Keyfactor Windows Orchestrator
  Change the schedule on these using PUT /CertificateAuthority (see PUT Certificate Authority on page 330).

Tip: The following permissions (see Security Overview) are required to use this feature:
AgentManagement: *Modify*
CertificateStoreManagement: *Schedule*

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 328: 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 700) 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 drop-downs for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide* Job History Search Feature section. Either a list of one or more *JobAuditIds* or a *Query* is **required**, but not both.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.19.6 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 482).
- SSH Synchronization
  Change the schedule on these using PUT /SSH/ServerGroups (see PUT SSH Server Groups on page 1027).

- SSL Discovery and Monitoring
  Change the schedule on these using PUT /SSL/Networks (see PUT SSL Networks on page 1146).
- CA Synchronization for Remote CAs Managed with the Keyfactor Universal Orchestrator or Keyfactor Windows Orchestrator
  Change the schedule on these using PUT /CertificateAuthority (see PUT Certificate Authority on page 330).

Tip: The following permissions (see Security Overview) are required to use this feature:
AgentManagement: *Modify*
CertificateStoreManagement: *Schedule*

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 329: POST Orchestrator Jobs Unschedule Input Parameters

Name	In	Description
JobIds	Body	**Required***. An array of GUIDs indicating the job IDs of the jobs that should be unscheduled. Use the *GET /OrchestratorJobs/ScheduledJobs* method (see GET Orchestrator Jobs Scheduled Jobs on page 705) 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, refer to the *Keyfactor Command Reference Guide* Orchestrator Scheduled Job Search Feature section. Either a list of one or more *JobIds* or a *Query* is **required**, but not both.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.19.7 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 Overview) are required to use this feature:
AgentManagement: *Modify*

Table 330: 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 700) with a query similar to the following to retrieve a list of all orchestrator jobs potentially eligible for acknowledgement: `(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 drop-downs for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide* Job History Search Feature section. Either a list of one or more *JobAuditIds* or a *Query* is **required**, but not both.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.19.8 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 Overview) are required to use this feature:
AgentManagement: *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 700) 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 699) to retrieve the data.

Table 331: POST Orchestrator Jobs Custom Bulk Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>Orches-tratorIds</td><td>Body</td><td>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 12) 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 12) 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.</td></tr>
<tr><td>JobTypeName</td><td>Body</td><td>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 565) to retrieve a list of your defined custom job types to determine the job type name to use.</td></tr>
<tr><td>Schedule</td><td>Body</td><td>An object containing the schedule for the custom job. 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>
For example, every hour:</td></tr>
</table>
</td></tr>
</table>

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

```
"Interval": {
   "Minutes": 60
}
```

</td></tr>
<tr><td>Daily</td><td>A dictionary that indicates a job scheduled to run every day at the same time with the parameter:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
```

</td></tr>
<tr><td>Weekly</td><td>A dictionary that indicates a job scheduled to run on a specific day or days every week at the same time with the parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</td><td>The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</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:

```
"Weekly": {
   "Days": [
      "Monday",
```

</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>

```
        "Wednesday",
        "Friday"
    ],
    "Time": "2022-02-27T17:30:00Z"
}
```

</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}
```

</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:

```
"ExactlyOnce": {
    "Time": "2022-02-27T11:45:00Z"
}
```

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 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 array of key/value pairs 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 565) 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 332: POST Orchestrator Jobs Custom Bulk Response Data

Name	Description
OrchestratorJobPairs	An array containing identifying information for each orchestrator on which the job will be run. Orchestrator job pair parameters include: **Value** / **Description** JobId — A string indicating the Keyfactor Command reference GUID for the job. OrchestratorId — A string indicating the Keyfactor Command reference GUID of the orchestrator that will execute this job.
JobTypeName	A string indicating the reference name for the custom job type for the job.
Schedule	An object containing the schedule for the custom job.
JobFields	An array of key/value pairs that set the values for any optional job fields configured for the custom job type.
RequestTimestmap	The date, in UTC, when the custom job was submitted.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.20 PAM Providers

Privileged Access Management (PAM) functionality in Keyfactor Web APIs allows for configuration of third party PAM providers to secure certificate stores. In the current release, both CyberArk and Delinea (formerly Thycotic) are supported. The PAM component of the Keyfactor API includes methods necessary to programmatically create, delete, edit, and list PAM Providers.

Table 333: PamProviders Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes a PAM provider.	DELETE PAM Providers ID on the next page
/{id}	GET	Returns information for the specified PAM provider.	GET PAM Providers ID on the next page
/Types	GET	Returns a list of all available PAM provider types.	GET PAM Providers Types on page 733

Endpoint	Method	Description	Link
/Types	POST	Creates a new PAM provider type.	POST PAM Providers Types on page 736
/	GET	Returns a list of all the configured PAM providers.	GET PAM Providers on page 739
/	POST	Creates a new PAM provider.	POST PAM Providers on page 748
/	PUT	Updates a PAM provider.	PUT PAM Providers on page 764

2.2.20.1 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 Overview) are required to use this feature:
PrivilegedAccessManagement: *Modify*

Table 334: DELETE PamProviders {id} 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 739) to retrieve a list of all the PAM providers to determine the PAM provider's ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.20.2 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 Overview) are required to use this feature:
PrivilegedAccessManagement: *Read*
SystemSettings: *Read*

Table 335: GET PamProviders {id} 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 739) to retrieve a list of all the PAM providers to determine the provider's ID.

Table 336: GET PamProviders {id} 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 used to identify the PAM provider throughout Keyfactor.
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 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 parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new 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.

Name	Description
	Value / **Description**

Value	Description
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).  **Example:** For CyberArk when defining a PAM provider, you configure two CyberArk-specific fields: • PrivateArk Safe: The name of the safe in CyberArk containing the certificate store password you wish to use. • Application ID: The name of the application created in CyberArk for use with Keyfactor Command. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so: `{ "Id": 1, "Name": "Safe", "DisplayName": "Priv-ateArk Safe", "DataType": 1, "InstanceLevel": false,`

Name	Description
	Value / **Description**
	Value / **Description**

```
   "ProviderType": null
},
{
   "Id": 4,
   "Name": "AppId",
   "DisplayName": "Applic-
ation ID",
   "DataType": 1,
   "InstanceLevel": false,
   "ProviderType": null
}
```

When you configure a certificate store to use CyberArk as a credential provider, you enter the name of the folder in the CyberArk safe where the protected object is stored and you enter the name of the projected object in the CyberArk safe 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:

```
{
   "Id": 2,
   "Name": "Folder",
   "DisplayName": "Priv-
ateArk Folder Name",
   "DataType": 1,
   "InstanceLevel": true,
   "ProviderType": null
},
{
   "Id": 3,
   "Name": "Object",
```

<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> 
<pre>
   "DisplayName": "Priv-
ateArk Protected Password
Name",
   "DataType": 1,
   "InstanceLevel": true,
   "ProviderType": null
}
</pre>
In both cases, the values for the fields (e.g. the actual name of the object in CyberArk where the password is stored) are stored in the ProviderTypeParamValues array.</td></tr>
<tr><td>ProviderType</td><td>An array containing details for the provider type. Provider type parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID 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>
</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>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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Provider-TypeParamValues</td><td>An array containing the values for the provider types specified by ProviderTypeParams. 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 user-name 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>
<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.</td></tr>
<tr><td>Provider</td><td>An array containing information about the provider.</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>Provider-TypeParams</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. 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 PAM provider. For parameters with an InstanceLevel of True, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are:
1 = String
2 = Secret</td></tr>
<tr><td>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>
<tr><td>ProviderType</td><td>An array containing details for the provider type. Provider type parameters include:</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>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID 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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</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.
You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting pass-</td></tr>
</table>

Name	Description
	words for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.20.3 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.

Tip: The following permissions (see Security Overview) are required to use this feature:
PrivilegedAccessManagement: *Read*
SystemSettings: *Read*

Table 337: GET PamProviders Types 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.

Name	Description
ProviderTypeParams	An array containing parameters set for the PAM provider type.

Value	Description
Id	An integer indicating the ID of the type. Possible values are:

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 provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.
DataType	An integer indicating the data type for the parameter. Possible values are: • 1 = String • 2 = Secret
InstanceLevel	A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (True). **Example:** For CyberArk when defining a PAM provider, you configure two CyberArk-specific fields: • PrivateArk Safe: The name of the safe in CyberArk containing the certificate store password you wish to use. • Application ID: The name of the application created in CyberArk for use with Keyfactor Command.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.20.4 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 Overview) are required to use this feature:
PrivilegedAccessManagement: *Modify*
SystemSettings: *Read*

Table 338: POST PamProviders Types Input Parameters

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Name</td><td>A string containing the name of the PAM provider type.</td></tr>
<tr><td>Parameters</td><td>An array containing parameters for the provider type. Parameter details include:
<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 parameter.</td></tr>
<tr><td>DisplayName</td><td>A string indicating the display name for the PAM provider type parameter. For parameters with an InstanceLevel of False, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an InstanceLevel of True, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are:
1 = String
2 = Secret</td></tr>
<tr><td>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).

  Example: For CyberArk when defining a PAM provider, you configure two CyberArk-specific fields:
PrivateArk Safe: The name of the safe in CyberArk containing the certificate store password you wish to use.
Application ID: The name of the application created in CyberArk for use with Keyfactor Command.
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

```
{
   "Id": 1,
   "Name": "Safe",
   "DisplayName": "PrivateArk
Safe",
   "DataType": 1,
   "InstanceLevel": false
   "ProviderType": null
},
{
   "Id": 4,
   "Name": "AppId",
   "DisplayName": "Application
ID",
   "DataType": 1,
   "InstanceLevel": false
   "ProviderType": null
}
```

When you configure a certificate store to use CyberArk as a credential provider, you enter the name of the folder in the CyberArk safe where the protected object is stored and you enter the name of the projected object in the CyberArk safe 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:

```
{
   "Id": 2,
   "Name": "Folder",
   "DisplayName": "PrivateArk
Folder Name",
   "DataType": 1,
   "InstanceLevel": true
   "ProviderType": null
},
{
   "Id": 3,
   "Name": "Object",
   "DisplayName": "PrivateArk
```

Name	Description
	Value / **Description**

```
Protected Password Name",
   "DataType": 1,
   "InstanceLevel": true
   "ProviderType": null
}
```

In both cases, the *values* for the fields (e.g. the actual name of the object in CyberArk where the password is stored) are stored in the *ProviderTypeParamValues* array.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.20.5 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 Overview) are required to use this feature:
PrivilegedAccessManagement: *Read*
SystemSettings: *Read*

Table 339: GET PamProviders 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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 340: GET PamProviders 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 used to identify the PAM provider throughout Keyfactor.
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 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 parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new 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.

Name	Description
	Value / **Description**

Value	Description
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).  **Example:** For CyberArk when defining a PAM provider, you configure two CyberArk-specific fields: • PrivateArk Safe: The name of the safe in CyberArk containing the certificate store password you wish to use. • Application ID: The name of the application created in CyberArk for use with Keyfactor Command. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so: `{ "Id": 1, "Name": "Safe", "DisplayName": "Priv-ateArk Safe", "DataType": 1, "InstanceLevel": false,`

Name	Description
	Value / **Description**
	Value / **Description**

```
   "ProviderType": null
},
{
   "Id": 4,
   "Name": "AppId",
   "DisplayName": "Applic-
ation ID",
   "DataType": 1,
   "InstanceLevel": false,
   "ProviderType": null
}
```

When you configure a certificate store to use CyberArk as a credential provider, you enter the name of the folder in the CyberArk safe where the protected object is stored and you enter the name of the projected object in the CyberArk safe 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:

```
{
   "Id": 2,
   "Name": "Folder",
   "DisplayName": "Priv-
ateArk Folder Name",
   "DataType": 1,
   "InstanceLevel": true,
   "ProviderType": null
},
{
   "Id": 3,
   "Name": "Object",
```

Name	Description
	Value / **Description** **Value** / **Description**  `"DisplayName": "PrivateArk Protected Password Name", "DataType": 1, "InstanceLevel": true, "ProviderType": null }` In both cases, the *values* for the fields (e.g. the actual name of the object in CyberArk where the password is stored) are stored in the *ProviderTypeParamValues* array. ProviderType — An array containing details for the provider type. Provider type parameters include: **Value** / **Description** Id — The Keyfactor Command reference GUID for the PAM provider type parameter. Name — A string indicating the internal name 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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Provider-TypeParamValues</td><td>An array containing the values for the provider types specified by ProviderTypeParams. 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 user-name 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>
<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.</td></tr>
<tr><td>Provider</td><td>An array containing information about the provider.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Value / **Description** Provider-TypeParams — An array of parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new PAM provider. For parameters with an *InstanceLevel* of *True*, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.
DataType	An integer indicating the data type for the parameter. Possible values are: • 1 = String • 2 = Secret
InstanceLevel	A Boolean that sets whether the parameter is used to define the underlying PAM provider (False) or a field that needs to be set to a value when configuring a certificate store to use the PAM provider (True). See example, above.
ProviderType	An array containing details for the provider type. Provider type parameters include:

<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>Id</td><td>The Keyfactor Command reference GUID 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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</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.
You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting pass-</td></tr>
</table>

Name	Description
	words for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.20.6 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 Overview) are required to use this feature:
CertificateStoreManagement: *Modify*
PrivilegedAccessManagement: *Modify*
SystemSettings: *Read*

Table 341: POST PamProviders 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 used to identify the PAM provider throughout Keyfactor.</td></tr>
<tr><td>Area</td><td>Body</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>ProviderType</td><td>Body</td><td>An array containing details about the provider type for the provider. Provider type details include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID for the provider type.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the provider type.</td></tr>
<tr><td>Provider-TypeParams</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. 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>DisplayNam-e</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>
</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>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></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 certificate store to use the PAM provider (True).

  Example: For CyberArk when defining a PAM provider, you configure two CyberArk-specific fields:
PrivateArk Safe: The name of the safe in CyberArk containing the certificate store password you wish to use.
Application ID: The name of the application created in CyberArk for use with Keyfactor Command.
Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:

<pre>{
  "Id": 1,
  "Name": "Safe",
  "DisplayName":</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Value / **Description** **Value** / **Description**  `"PrivateArk Safe", "DataType": 1, "InstanceLevel": false, "ProviderType": null }, { "Id": 4, "Name": "AppId", "DisplayName": "Application ID", "DataType": 1, "InstanceLevel": false, "ProviderType": null }` When you configure a certificate store to use CyberArk as a credential provider, you enter the name of the folder in the CyberArk safe where the protected object is stored and you enter the name of the projected object in the CyberArk safe 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: `{ "Id": 2,`

<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> 
<pre>
   "Name": "Folder",
   "DisplayName":
"PrivateArk Folder
Name",
   "DataType": 1,
   "InstanceLevel":
true,
   "ProviderType":
null
},
{
   "Id": 3,
   "Name": "Object",
   "DisplayName":
"PrivateArk Protected
Password Name",
   "DataType": 1,
   "InstanceLevel":
true,
   "ProviderType":
null
}
</pre>
In both cases, the values for the fields (e.g. the actual name of the object in CyberArk where the password is stored) are stored in the Provider-TypeParamValues array.</td></tr>
<tr><td>Provider-Type</td><td>An array containing details for the provider type. Provider type parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor</td></tr>
</table>
</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>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>Command reference GUID 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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>Body</td><td>An array containing the values for the provider types specified by ProviderTypeParams. Provider type parameter values 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>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 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>
<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.</td></tr>
<tr><td>Provider</td><td>An array containing information about the provider.</td></tr>
<tr><td>Provider-TypeParams</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. 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>DisplayNam-e</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</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>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>creates a new PAM provider.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are:
1 = String
2 = Secret</td></tr>
<tr><td>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>
<tr><td>ProviderType</td><td>An array containing details for the provider type. Provider type parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID 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>
</table>
</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>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>Provider-TypeParams</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>
<tr><td>SecureAreaId</td><td>Body</td><td>An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any.
You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting passwords for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers.</td></tr>
</table>

Table 342: POST PamProviders 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 used to identify the PAM provider throughout Keyfactor.
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 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 parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new 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.

Name	Description
	Value / **Description**

Value	Description
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).  **Example:** For CyberArk when defining a PAM provider, you configure two CyberArk-specific fields: • PrivateArk Safe: The name of the safe in CyberArk containing the certificate store password you wish to use. • Application ID: The name of the application created in CyberArk for use with Keyfactor Command. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so: `{ "Id": 1, "Name": "Safe", "DisplayName": "Priv-ateArk Safe", "DataType": 1, "InstanceLevel": false,`

Name	Description
	Value / **Description**
	Value / **Description**

```
   "ProviderType": null
},
{
   "Id": 4,
   "Name": "AppId",
   "DisplayName": "Applic-
ation ID",
   "DataType": 1,
   "InstanceLevel": false,
   "ProviderType": null
}
```

When you configure a certificate store to use CyberArk as a credential provider, you enter the name of the folder in the CyberArk safe where the protected object is stored and you enter the name of the projected object in the CyberArk safe 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:

```
{
   "Id": 2,
   "Name": "Folder",
   "DisplayName": "Priv-
ateArk Folder Name",
   "DataType": 1,
   "InstanceLevel": true,
   "ProviderType": null
},
{
   "Id": 3,
   "Name": "Object",
```

<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> 
<pre>
   "DisplayName": "Priv-
ateArk Protected Password
Name",
   "DataType": 1,
   "InstanceLevel": true,
   "ProviderType": null
}
</pre>
In both cases, the values for the fields (e.g. the actual name of the object in CyberArk where the password is stored) are stored in the ProviderTypeParamValues array.</td></tr>
<tr><td>ProviderType</td><td>An array containing details for the provider type. Provider type parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID 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>
</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>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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Provider-TypeParamValues</td><td>An array containing the values for the provider types specified by ProviderTypeParams. 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 user-name 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>
<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.</td></tr>
<tr><td>Provider</td><td>An array containing information about the provider.</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>Provider-TypeParams</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. 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 PAM provider. For parameters with an InstanceLevel of True, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are:
1 = String
2 = Secret</td></tr>
<tr><td>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>
<tr><td>ProviderType</td><td>An array containing details for the provider type. Provider type parameters include:</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>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID 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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</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.
You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting pass-</td></tr>
</table>

Name	Description
	words for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.20.7 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 Overview) are required to use this feature:
CertificateStoreManagement: *Modify*
PrivilegedAccessManagement: *Modify*
SystemSettings: *Read*

Warning: 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 343: PUT PamProviders 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 used to identify the PAM provider throughout Keyfactor.</td></tr>
<tr><td>Area</td><td>Body</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>ProviderType</td><td>Body</td><td>An array containing details about the provider type for the provider.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>A string indicating the Keyfactor Command reference GUID for the provider type.</td></tr>
<tr><td>Name</td><td>A string indicating the name of the provider type.</td></tr>
<tr><td>Provider-TypeParams</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.
<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>DisplayNam-e</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>
</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>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></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 certificate store to use the PAM provider (True).

  Example: For CyberArk when defining a PAM provider, you configure two CyberArk-specific fields:
PrivateArk Safe: The name of the safe in CyberArk containing the certificate store password you wish to use.
Application ID: The name of the application created in CyberArk for use with Keyfactor Command.
Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so:

<pre>{
   "Id": 1,
   "Name": "Safe",
   "DisplayName":</pre></td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Value / **Description** **Value** / **Description**  `"PrivateArk Safe", "DataType": 1, "InstanceLevel": false, "ProviderType": null }, { "Id": 4, "Name": "AppId", "DisplayName": "Application ID", "DataType": 1, "InstanceLevel": false, "ProviderType": null }` When you configure a certificate store to use CyberArk as a credential provider, you enter the name of the folder in the CyberArk safe where the protected object is stored and you enter the name of the projected object in the CyberArk safe 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: `{ "Id": 2,`

<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> 
<pre>
   "Name": "Folder",
   "DisplayName":
"PrivateArk Folder
Name",
   "DataType": 1,
   "InstanceLevel":
true,
   "ProviderType":
null
},
{
   "Id": 3,
   "Name": "Object",
   "DisplayName":
"PrivateArk Protected
Password Name",
   "DataType": 1,
   "InstanceLevel":
true,
   "ProviderType":
null
}
</pre>
In both cases, the values for the fields (e.g. the actual name of the object in CyberArk where the password is stored) are stored in the Provider-TypeParamValues array.</td></tr>
<tr><td>Provider-Type</td><td>An array containing details for the provider type.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command</td></tr>
</table>
</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>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>reference GUID 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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>ProviderTypeParamValues</td><td>Body</td><td>An array containing the values for the provider types specified by ProviderTypeParams.
<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>
</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>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 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 provider. If you are attaching to something with a GUID ID, this will be used.</td></tr>
<tr><td>Provider</td><td>An array containing information about the provider.</td></tr>
<tr><td>Provider-TypeParams</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.
<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>DisplayNam-e</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:</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>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>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 certificate store to use the PAM provider (True).
See example, above.</td></tr>
<tr><td>ProviderType</td><td>An array containing details for the provider type.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID 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>ProviderTypeParams</td><td>An array of parameters that the</td></tr>
</table>
</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>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>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>
<tr><td>SecureAreaId</td><td>Body</td><td>An integer indicating the Keyfactor Command reference ID for the certificate store container the PAM provider is associated with, if any.
You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting passwords for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers.</td></tr>
</table>

Table 344: PUT PamProviders 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 used to identify the PAM provider throughout Keyfactor.
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 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 parameters that the provider type uses for data input in Keyfactor Command when creating new PAM provider and certificate store records. Provider type parameters values include:

Value	Description
Id	An integer indicating the Keyfactor Command reference ID for the PAM provider type parameter.
Name	A string indicating the internal name for the PAM provider type parameter.
DisplayName	A string indicating the display name for the PAM provider type parameter. For parameters with an *InstanceLevel* of *False*, this name appears on the PAM provider dialog for the parameter when a user creates a new 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.

Name	Description
	Value / **Description**

Value	Description
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).  **Example:** For CyberArk when defining a PAM provider, you configure two CyberArk-specific fields: • PrivateArk Safe: The name of the safe in CyberArk containing the certificate store password you wish to use. • Application ID: The name of the application created in CyberArk for use with Keyfactor Command. Because these fields are configured on the PAM provider definition, they appear as InstanceLevel=False like so: `{ "Id": 1, "Name": "Safe", "DisplayName": "Priv-ateArk Safe", "DataType": 1, "InstanceLevel": false,`

Name	Description
	Value / **Description**
	Value / **Description**

```
   "ProviderType": null
},
{
   "Id": 4,
   "Name": "AppId",
   "DisplayName": "Applic-
ation ID",
   "DataType": 1,
   "InstanceLevel": false,
   "ProviderType": null
}
```

When you configure a certificate store to use CyberArk as a credential provider, you enter the name of the folder in the CyberArk safe where the protected object is stored and you enter the name of the projected object in the CyberArk safe 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:

```
{
   "Id": 2,
   "Name": "Folder",
   "DisplayName": "Priv-
ateArk Folder Name",
   "DataType": 1,
   "InstanceLevel": true,
   "ProviderType": null
},
{
   "Id": 3,
   "Name": "Object",
```

<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> 
<pre>
   "DisplayName": "Priv-
ateArk Protected Password
Name",
   "DataType": 1,
   "InstanceLevel": true,
   "ProviderType": null
}
</pre>
In both cases, the values for the fields (e.g. the actual name of the object in CyberArk where the password is stored) are stored in the ProviderTypeParamValues array.</td></tr>
<tr><td>ProviderType</td><td>An array containing details for the provider type. Provider type parameters include:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID 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>
</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>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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Provider-TypeParamValues</td><td>An array containing the values for the provider types specified by ProviderTypeParams. 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 user-name 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>
<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.</td></tr>
<tr><td>Provider</td><td>An array containing information about the provider.</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>Provider-TypeParams</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. 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 PAM provider. For parameters with an InstanceLevel of True, this name appears on the Server dialog for the parameter when a user creates a new PAM provider.</td></tr>
<tr><td>DataType</td><td>An integer indicating the data type for the parameter. Possible values are:
1 = String
2 = Secret</td></tr>
<tr><td>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>
<tr><td>ProviderType</td><td>An array containing details for the provider type. Provider type parameters include:</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>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Id</td><td>The Keyfactor Command reference GUID 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>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.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>
</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.
You can create a single PAM provider for each provider type (e.g. CyberArk), however, if you have opted to organize your certificate stores into containers, you will need to create multiple providers to match your container organization structure. The container field in the PAM provider definition is not required, but if one is supplied when creating a PAM provider, the PAM provider can only be used with certificate stores in the matching container. Likewise, a PAM provider defined with no container would be available for selection when setting pass-</td></tr>
</table>

Name	Description
	words for any certificate store that also did not specify a container. A PAM provider configured in this way could be used across a variety of certificate stores (e.g. both JKS and F5) as long as they were not in containers.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21 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 345: 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 788
/Custom/{id}	GET	Returns the custom report with the specified ID.	GET Reports Custom ID on page 789
/Schedules/{id}	DELETE	Deletes the schedule for the built-in report with the specified schedule ID.	DELETE Reports Schedules ID on page 790
/Schedules/{id}	GET	Returns the schedule for the built-in report with the specified schedule ID.	GET Reports Schedules ID on page 790
/{id}/Parameters	GET	Returns the parameters for the built-in report with the specified report ID.	GET Reports ID Parameters on page 794
/{id}/Parameters	PUT	Updates the parameters for the built-in report with the specified report ID.	PUT Reports ID Parameters on page 795
/	GET	Returns all built-in reports with filtering and output options.	GET Reports on page 797
/	PUT	Updates the built-in report with the specified ID. Only some fields can be updated.	PUT Reports on page 800
/Custom	GET	Returns all custom reports with filtering and	GET Reports Custom on

Endpoint	Method	Description	Link
		output options.	page 803
/Custom	POST	Creates a custom report.	POST Reports Custom on page 805
/Custom	PUT	Updates the custom report with the specified ID.	PUT Reports Custom on page 807
/{id}/Schedules	GET	Returns the schedule for the built-in report with the specified report ID.	GET Reports ID Schedules on page 808
/{id}/Schedules	POST	Creates a schedule for the built-in report with the specified report ID.	POST Reports ID Schedules on page 812
/{id}/Schedules	PUT	Updates a schedule for the built-in report with the specified report ID.	PUT Reports ID Schedules on page 821

2.2.21.1 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 Overview) are required to use this feature:
Reports: *Read*

Table 346: 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 797) to retrieve a list of your built-in reports to determine the report ID to use.

Table 347: 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 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 365). This corresponds to the Keyfactor

Name	Description
	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 containing the parameters for the report. . Report parameters include:

Name	Description
Id	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.

<table>
<tr><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: 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 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>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>ReportFormat</td><td>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</td></tr>
<tr><td>KeyfactorSchedule</td><td>An array 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>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name / **Description**

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. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }`
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. 2021-05-19T16:23:01Z). 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>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>

```
"Weekly": {
   "Days": [
      "Monday",
      "Wednesday",
      "Friday"
   ],
   "Time": "2022-02-27T17:30:00Z"
}
```

</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}
```

</td></tr>
</table>
Note: Although the 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 containing the email addresses of users configured as recipients of the scheduled report, if any.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**
RuntimePara-meters	Any array containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:

Name	Description
CertAuth	The certificate authority or authorities selected to report on.
EndDate	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).
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.

<table>
<tr><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>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>Accep-tedSched-uleFormats</td><td>An array containing the report formats supported for the report. Typically supported formats are PDF and Excel. Select reports support CSV format.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.2 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 Overview) are required to use this feature:
Reports: *Modify*

Table 348: 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 803) to retrieve a list of your custom report links to determine the report ID to use.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.3 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 Overview) are required to use this feature:
Reports: *Read*

Table 349: 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.

Table 350: 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://mywebserver.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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.4 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 Overview) are required to use this feature:
Reports: *Modify*

Table 351: 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 797) 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 781) to retrieve the details for that report to determine the schedule ID to use.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.5 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 808).

Tip: The following permissions (see Security Overview) are required to use this feature:
Reports: *Read*

Table 352: 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 797) 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 781) to retrieve the details for that report to determine the schedule ID to use.

Table 353: GET Reports Schedules {id} Response Data

Name	Description
Id	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 array 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. 2021-05-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
```

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
	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. 2021-05-19T16:23:01Z).
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": "2022-02-27T17:30:00Z"
}
```

Monthly

A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).
Day	The number of the day, in the month, to run the job.

For example, on the first of every month at 5:30 pm:

```
"Monthly": {
   "Day": 1
   "Time": "2022-02-27T17:30:00Z"
}
```

Note: Although the Swagger *Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this

Name	Description
	functionality—are valid for this endpoint.
EmailRecipients	An array containing the email addresses of users configured as recipients of the scheduled report, if any.
RuntimePara-meters	Any array containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:

Name	Description
CertAuth	The certificate authority or authorities selected to report on.
EndDate	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).
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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.6 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 Overview) are required to use this feature:
Reports: *Read*

Table 354: 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 797) to retrieve a list of your built-in reports to determine the report ID to use.

Table 355: GET Reports {id} Parameters Response Data

Name	Description
Id	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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.7 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 Overview) are required to use this feature:
Reports: *Modify*

Table 356: 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 the next page) 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 781) 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 357: PUT Reports {id} Parameters Response Data

Name	Description
Id	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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.8 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 781).

Tip: The following permissions (see Security Overview) are required to use this feature:
Reports: *Read*

Table 358: GET Reports 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. The query fields supported for this endpoint are: • *Categories* (CertificateCounts, CertificateLifecycle, Certificate Locations, PKIOperations, SecurityVulnerability,SSHKeys) • *Custom* • *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 359: 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). **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.

Name	Description
	Certificate de-duping is configured on a certificate collection using the *DuplicationField* parameter (see POST Certificate Collections on page 365). 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.9 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 812) or *PUT /Reports/{id}Schedules* (see PUT Reports ID Schedules on page 821) method. To update parameters for a built-in report, use the *PUT /Reports/{id}/Parameters* method (see PUT Reports ID Parameters on page 795). This method returns HTTP 200 OK on a success with the details of the report.

Tip: The following permissions (see Security Overview) are required to use this feature:
Reports: *Modify*

Table 360: 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 797) 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 365). 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 361: 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 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*

Name	Description
	parameter (see POST Certificate Collections on page 365). 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.10 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 Overview) are required to use this feature:
Reports: *Read*

Table 362: GET Reports Custom 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. The query fields supported for this endpoint are: • *Categories* (CertificateCounts, CertificateLifecycle, Certificate Locations, PKIOperations, SecurityVulnerability,SSHKeys) • *Custom* • *Favorite* (true, false) • *InNavigator* (true, false) • *Scheduled* (Number of schedules)
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 363: 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://mywebserver.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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.11 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 Overview) are required to use this feature:
Reports: *Modify*

Table 364: 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 365: 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://mywebserver.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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.12 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 Overview) are required to use this feature:
Reports: *Modify*

Table 366: 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 803) 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 367: 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://mywebserver.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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.13 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 790).

Tip: The following permissions (see Security Overview) are required to use this feature:
Reports: *Read*

Table 368: 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 797) to retrieve a list of your built-in reports to determine the report ID to use.

Table 369: GET Reports {id} Schedules Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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>ReportFormat</td><td>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</td></tr>
<tr><td>KeyfactorSchedule</td><td>An array 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).
	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": "2022-02-27T17:30:00Z"
}
```

Monthly — A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).
Day	The number of the day, in the month, to run the job.

For example, on the first of every month at 5:30 pm:

```
"Monthly": {
   "Day": 1
   "Time": "2022-02-27T17:30:00Z"
}
```

Note: Although the Swagger *Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this

Name	Description
	functionality—are valid for this endpoint.
EmailRecipients	An array containing the email addresses of users configured as recipients of the scheduled report, if any.
RuntimePara-meters	Any array containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:

Name	Description
CertAuth	The certificate authority or authorities selected to report on.
EndDate	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).
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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.14 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 Overview) are required to use this feature:
Reports: *Modify*

Table 370: POST 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 797) 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 array 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:

<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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}
</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>

Note: Although the 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>"KeyfactorSchedule": {
   "Monthly": {
      "Day": 1,
      "Time": "2021-07-01T17:00:00Z"
   }
},</pre>
Or:
<pre>"KeyfactorSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Thursday"
      ],
      "Time": "2021-07-01T17:00:00Z"
   }
},</pre>
</td></tr>
<tr><td>EmailRecipients</td><td>Body</td><td>Required*. An array containing the email addresses of users configured as recipients of the</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>scheduled report, if any. For example:

<pre>"EmailRecipients": [
   "pkiadmins@keyexample.com",
   "john.smith@keyexample.com"
]</pre>
This field is required if SendReport is set to true.</td></tr>
<tr><td>RuntimeParameters</td><td>Body</td><td>Required*. Any array 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>
<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.</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>This is configured as a certain number of days, weeks or months before or after the current date (e.g. 30 days before today).</td></tr><tr><td>TemplateIds</td><td>The Keyfactor Command identifiers for the templates to include in the report.</td></tr></table> For example: <pre>"RuntimeParameters": { "StartDate": "60-Day-Before", "EndDate": "7-Day-Before", "Metadata": "AppOwnerFirstName, AppOwnerLastName", "Requesters": "jsmith" }</pre> This field is **required** for reports that have runtime parameters.

Table 371: POST Reports {id} Schedules Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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>ReportFormat</td><td>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</td></tr>
<tr><td>KeyfactorSchedule</td><td>An array 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).
	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": "2022-02-27T17:30:00Z"
}
```

Monthly — A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).
Day	The number of the day, in the month, to run the job.

For example, on the first of every month at 5:30 pm:

```
"Monthly": {
   "Day": 1
   "Time": "2022-02-27T17:30:00Z"
}
```

Note: Although the Swagger *Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this

Name	Description
	functionality—are valid for this endpoint.
EmailRecipients	An array containing the email addresses of users configured as recipients of the scheduled report, if any.
RuntimePara-meters	Any array containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:

Name	Description
CertAuth	The certificate authority or authorities selected to report on.
EndDate	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).
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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.21.15 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 Overview) are required to use this feature:
Reports: *Modify*

Table 372: PUT 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 797) 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 **schedule**. Use the *GET /Reports/{id}* (see GET Reports ID on page 781) 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 array providing the schedule for the report. **Name** / **Description** Off — Turn off a previously configured schedule.

<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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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>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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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>"KeyfactorSchedule": {
   "Monthly": {
      "Day": 1,
      "Time": "2021-07-01T17:00:00Z"
   }
},</pre>
Or:
<pre>"KeyfactorSchedule": {
   "Weekly": {
      "Days": [
         "Monday",
         "Thursday"
      ],
      "Time": "2021-07-01T17:00:00Z"
   }</pre>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td><pre>},</pre></td></tr>
<tr><td>EmailRecipients</td><td>Body</td><td>Required*. An array containing the email addresses of users configured as recipients of the scheduled report, if any. For example:<pre>"EmailRecipients": [
   "pkiadmins@keyexample.com",
   "john.smith@keyexample.com"
]</pre>This field is required if SendReport is set to true.</td></tr>
<tr><td>RuntimePara-meters</td><td>Body</td><td>Required*. Any array 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 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>

Name	In	Description
		<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> For example: <pre>"RuntimeParameters": { "StartDate": "60-Day-Before", "EndDate": "7-Day-Before", "Metadata": "AppOwnerFirstName, AppOwnerLastName", "Requesters": "jsmith" }</pre> This field is **required** for reports that have runtime parameters.

Table 373: PUT Reports {id} Schedules Response Data

Name	Description
Id	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 array 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. 2021-05-19T16:23:01Z).

For example, daily at 11:30 pm:

```
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
```

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

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. 2021-05-19T16:23:01Z).
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": "2022-02-27T17:30:00Z"
}
```

Monthly

A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).
Day	The number of the day, in the month, to run the job.

For example, on the first of every month at 5:30 pm:

```
"Monthly": {
   "Day": 1
   "Time": "2022-02-27T17:30:00Z"
}
```

Note: Although the Swagger *Example Value* may show examples of various other schedules, only the schedules shown here—that are available in the Management Portal for this

Name	Description
	functionality—are valid for this endpoint.
EmailRecipients	An array containing the email addresses of users configured as recipients of the scheduled report, if any.
RuntimePara-meters	Any array containing the parameters to be used at run time configured in the report schedule. Runtime parameters will vary depending on the report selected. Runtime parameters may include things such as:

Name	Description
CertAuth	The certificate authority or authorities selected to report on.
EndDate	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).
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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.22 Security Identities

The Security Identities component of the Keyfactor API includes methods necessary to list, add, and delete security identities. The permissions set with these methods are used to control access to all aspects of Keyfactor Command.

Table 374: Security Identities Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the security identity with the specified ID.	DELETE Security Identities ID below
/{id}	GET	Returns permission details for the security identity with the specified ID.	GET Security Identities ID on the next page
/Lookup	GET	Validates that the identity with the specified name exists.	GET Security Identities Lookup on page 834
/	GET	Returns all security identities with filtering and output options.	GET Security Identities on page 835
/	POST	Adds a new security identity into Keyfactor Command.	POST Security Identities on page 854

2.2.22.1 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 835) to determine the ID of the security identity you wish to delete. This endpoint returns 204 with no content upon success.

Tip: The following permissions (see Security Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 375: DELETE Security Identities {id} Input Parameters

Name	In	Description
id	Path	**Required**. The ID of the security identity that should be deleted from Keyfactor Command.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.22.2 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.

Tip: The following permissions (see Security Overview) are required to use this feature:
SecuritySettings: *Read*

Table 376: 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 835) to retrieve a list of all the security identities to determine the identity's ID.

Table 377: 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 object containing a series of arrays with 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 the Security Roles and Identities page in the Keyfactor Command Reference Guide.</td></tr>
<tr><td>CollectionPermissions</td><td>An object containing information about the certificate collection permissions granted to the security identity. Collection 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</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>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 the Certificate Permissions page in the Keyfactor Command Reference Guide.
</td></tr>
<tr><td>ContainerPermissions</td><td>
An object containing information about the global 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>
For example:
<pre>
"ContainerPermissions": [
   {
      "Permission": "IIS Personal:CertificateStoreManagement_Read",
</pre>
</td></tr>
</table>

Name	Description
	<pre>"GrantedByRoles": ["Power Users", "Staff"] }, { "Permission": "F5 SSL Profiles REST:CertificateStoreManagement_ Schedule", "GrantedByRoles": ["Power Users"] },]</pre> For more information about container permissions, see the Container Permissions page in the *Keyfactor Command Reference Guide*.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.22.3 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 854) 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 Overview) are required to use this feature:
SecuritySettings: *Read*

Table 378: 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 379: GET Security Identities Lookup Response Data

Name	Description
Valid	A Boolean that indicates whether the provided name is valid (true) or not (false).

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.22.4 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.

Tip: The following permissions (see Security Overview) are required to use this feature:
SecuritySettings: *Read*

Table 380: GET Security Identities 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		*Not used.*
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 381: GET Security Identities 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 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>Roles</td><td>An array containing information about the security roles assigned to the security identity. Role information includes:
<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 GET Security Roles on page 914) 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>Immutable</td><td>Body</td><td>A Boolean that indicates whether the security role has been marked as editable (true) or not (false). Internal Keyfactor Command roles are not editable. This setting is reserved for Keyfactor Command internal use.</td></tr>
<tr><td>Valid</td><td>Body</td><td>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.</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>Identities</td><td>Body</td><td>An array containing information about the security identities assigned to the security role. Identity details include:</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>
<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 object containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. Possible values are:
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>AdminPortal
(a.k.a. Management Portal)</td><td>Read</td><td>Users can access the Management Portal. This permission must be enabled for all roles that will access the Management Portal.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Read</td><td>Users can view the agent auto-registration settings; Users must also have Read permissions for Agent Management.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Modify</td><td>Users can modify the agent auto-registration settings.</td></tr>
<tr><td>AgentManagement</td><td>Read</td><td>Users can access the Management Portal areas</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **In** / **Description**

Name	Value	Description
		and API endpoints to: • View orchestrators, including filtering the orchestrator management grid • View orchestrator jobs, including status, schedules, failures and warnings
AgentManagement	Modify	Users can access the Management Portal areas and API endpoints to: • Manage orchestrators, including approving and disapproving them • Unschedule and reschedule orchestrator jobs
API	Read	Users can call the Classic (CMS) API endpoints.
ApplicationSettings	Read	Users can view the application settings.

Name	Description
	Name / **In** / **Description**

Name	Value	Description
ApplicationSettings	Modify	Users can modify the application settings.
Auditing	Read	Users can access the Audit Log page in the Management Portal, and will be able to make API requests to obtain data from the audit log (query, etc.). The System Settings drop-down menu will display the Audit Log option to users with the *Auditing* Read permission.
CertificateCollections	Modify	Users can add or edit certificate collections. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information.
CertificateEnrollment	EnrollPFX	Users can use the PFX Enrollment page in the Management Portal and use the PFX enrollment related API endpoints.
CertificateEnrollment	EnrollCSR	Users can use the CSR Enrollment page in the Management Portal and use the CSR enrollment related API endpoints.
CertificateEnrollment	CsrGeneration	Users can use the CSR Generation page in the Management Portal and

Name	Description
	Name / **In** / **Description**

Name	Value	Description
		use the CSR generation related API endpoints.
CertificateEnrollment	PendingCsr	Users can use manage pending CSRs.
CertificateMetadataTypes	Read	Users can read custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints.
CertificateMetadataTypes	Modify	Users can add, edit, and delete custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints.
CertificateStoreManagement	Read	Users can view certificate stores—including the stores and containers but not discovery records—and certificate store types. Users who also have Read permissions for *Certificates* can view inventory for a certificate store. See *Container Permissions* in the *Keyfactor Command Reference Guide* for more information.

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** CertificateStoreManagement / Modify / Users can manage certificate stores—including the stores, containers, and discovery process—and certificate store types. Note that this permission does not control additions of certificates to certificate stores. CertificateStoreManagement / Schedule / Users can add certificates to certificate stores, renew/reissue certificates, and remove certificates from certificate stores. Certificates / Read / Users can view certificates in certificate search and certificate collections in the Management Portal and with related API endpoints, including certificate history, and can download certificates. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information.

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** Certificates / Import / Users can import certificates through Add Certificate in the Management Portal and with related API endpoints. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores from Add Certificate. Certificates / Recover / Users can download the certificates with their private key. Certificates / Revoke / Users can revoke certificates through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints. Certificates / Delete / Users can delete certificates and, if applicable, the private keys of the certificates from the Keyfactor Command database. Certificates / ImportPrivateKey / Users can save the private key for the certificate in the Keyfactor Command database. Certificates / EditMetadata / Users can modify certificate metadata for certi-

Name	Description
	Name / **In** / **Description**

Name	Value	Description
		ficates accessed through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints.
Dashboard	Read	Users can view the panels on their personalized dashboard and add and remove them.
Dashboard	RiskHeader	Users can view the risk header at the top of the dashboard.
EventHandlerRegistration	Read	Users can view the event handler registration settings.
EventHandlerRegistration	Modify	Users can modify the event handler registration settings.
MacAutoEnrollManagement	Read	Users can view the Mac Auto-Enroll Management settings.
MacAutoEnrollManagement	Modify	Users can modify the Mac Auto-Enroll Management settings.
Monitoring	Read	Users can view the expiration alerts in the Certificate Alerts in the Management Portal and with related API endpoints, including the alert schedule.

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** Monitoring / Modify / Users can modify the expiration alerts, including the alert text, recipients and event handlers. Users can also add new alerts, delete alerts and configure the expiration alert delivery schedule. Monitoring / Test / Users can test the expiration alerts, including sending email to recipients. Users must also have Read permissions for *Monitoring*. PkiManagement / Read / Users can view the Keyfactor Command PKI management settings within the following Management Portal areas and use related endpoints: • Certificate Authorities • Certificate Templates • Revocation Monitoring PkiManagement / Modify / Users can modify the Keyfactor Command PKI management settings: • Import, add, edit, and delete certi-

<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>
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td></td><td></td><td>ficate authorities
• Import certificate templates
• Add, edit, delete, and test revocation monitoring endpoints
• Configure revocation monitoring schedule
• Configure revocation monitoring recipients</td></tr>
<tr><td>PrivilegedAccessManagement</td><td>Read</td><td>Users can view PAM providers.</td></tr>
<tr><td>PrivilegedAccessManagement</td><td>Modify</td><td>Users can add, edit, and delete PAM providers.</td></tr>
<tr><td>Reports</td><td>Read</td><td>Users can generate and view reports.</td></tr>
<tr><td>Reports</td><td>Modify</td><td>Users can modify the delivery schedule for reports in Report Manager in the Management Portal and add, edit, and delete custom reports.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** **Note:** Report scheduling is limited by collection permissions. Users in roles that have *Reports: Read and Modify* permissions will also need to have *Read* collection permissions on individual collections to have the ability to add, edit and delete schedules associated with collections. The user will not have access to add, edit and delete schedules for any collections for which they do not have collection *Read* permissions in addition to *Reports* permissions. SecuritySettings / Read / Users can view the settings for Security Roles and Security Identities. Users must also have the Read permission for

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** *System Settings*. SecuritySettings / Modify / Users can modify the settings for Security Roles and Security Identities in the Management Portal and with related API endpoints. SSH / User / Users can generate their own SSH keys. SSH / ServerAdmin / Users can use all SSH functions, except creating server groups and assigning server group owners. Users have limited access to some functions based on server group ownership. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information. SSH / EnterpriseAdmin / Users can use all SSH functions. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information. SslManagement / Read / Users can view the SSL Network Discovery and Monitoring area in the Management Portal and with related

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** API endpoints, including defined networks and the network ranges configured for them, agent pools, and scan results. Users can use the query tool on the Results tab to find discovered endpoints and then view the discovered endpoints, including the details for the endpoints. SslManagement / Modify / Users can modify the SSL Network Discovery and Monitoring settings: • Create, edit, and delete networks, including scan schedules and notification recipients • Add, edit, and delete network ranges for networks • Add, edit, and delete agent pools • Add and remove discovered endpoints from monit-

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** oring SystemSettings / Read / Users can view the System Settings for: • Application Settings • Event Handler Registration to view built-in or custom event handlers • API Applications allowed to use the APIs for certificate lifecycle management • SMTP Configuration for email delivery of reports and alerts • Installed components • Licensing • Alerts and Warnings about the health of the Keyfactor Command system

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** SystemSettings / Modify / Users can modify the System Settings for: • Application Settings to configure many options for Keyfactor Command • Event Handler Registration to add or remove built-in or custom event handlers • Update SMTP Configuration for email delivery of reports and alerts • Installed components, including removing servers from use • Licensing, including the option to replace the existing license file WorkflowDefinitions / Read / Users can view the configured workflow definitions.

Name	Description
	Name / **In** / **Description** **Name** / **Value** / **Description** WorkflowDefinitions / Modify / Users can modify both the built-in and any custom workflow definitions, including the name and description and the configuration for the steps. Users can also add new workflow definitions, delete workflow definitions, publish workflow definitions, and import and export workflow definitions. WorkflowInstances / Manage / Users can manage initiated workflow instances, including stopping, restarting, and deleting them. WorkflowInstances / ReadAssignedToMe / Users can view the workflow instances that have been initiated and are awaiting input from them. **Tip:** There is not a security permission at this level that controls whether users can provide input (a signal) to a workflow instance. This is controlled using

Name	Description
	Name / **In** / **Description**

Name	Value	Description
		the security roles configured on the specific workflow definition. Any user who holds one of the roles configured in the workflow step that requires a signal may provide the necessary input. The user does not need to hold the *ReadAssignedToMe WorkflowInstances* permission in order to provide the input.
WorkflowInstances	ReadAll	Users can view all the workflow instances that have been initiated.
WorkflowInstances	ReadMy	Users can view the workflow instances that have been initiated by them (e.g. because they enrolled for a certificate).
WorkflowManagement (a.k.a. Alerts)	Read	Users can view the pending, issued, and denied workflow alerts.
WorkflowManagement	Modify	Users can modify the

<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>
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>(a.k.a. Alerts)</td><td></td><td>pending, issued, and denied workflow alerts, including the alert text, recipients, and event handlers. Users can also add new alerts, delete alerts, and configure the pending alert delivery schedule.</td></tr>
<tr><td>WorkflowManagement (a.k.a. Alerts)</td><td>Test</td><td>Users can test the pending alerts, including sending email to recipients. Users must also have Read permissions for Workflow.</td></tr>
<tr><td>WorkflowManagement (a.k.a. Certificate Requests)</td><td>Participate</td><td>Users can participate in the pending, issued and denied workflow process by approving or denying certificate requests from the Certificate Requests page or from the individual pages reached from links included in alerts in the Management Portal and with related API endpoints.</td></tr>
</table>
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.22.5 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 834) 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.

Tip: The following permissions (see Security Overview) are required to use this feature:
SecuritySettings: *Modify*

Tip: This method cannot be used to assign roles to an identity. Use the *PUT /Security/Roles* method (see PUT Security Roles on page 933) to assign roles to an identity.

Table 382: 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 383: 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23 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, container and collection-level.

Table 384: 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 858
/{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 859
/{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 879
/{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 900
/{id}/Permisssions/Containers	POST	Adds container permissions to the security role that matches the ID.	POST Security Roles ID Permissions Containers on page 901
/{id}/Permisssions/Containers	PUT	Sets container permissions to the security role that matches the ID.	PUT Security Roles ID Permissions Containers on page 903
/{id}/Permisssions/Collections	GET	Returns all collection permissions associated with the security role that matches the ID.	GET Security Roles ID Permissions Collections on

Endpoint	Method	Description	Link
			page 904
/{id}/Permisssions/Collections	POST	Adds collection permissions to the security role that matches the ID.	POST Security Roles ID Permissions Collections on page 905
/{id}/Permisssions/Collections	PUT	Sets collection permissions to the security role that matches the ID.	PUT Security Roles ID Permissions Collections on page 906

2.2.23.1 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 Overview) are required to use this feature:
SecuritySettings: *Read*

Table 385: GET Security Roles {id} Permissions Input Parameters

Name	In	Description
id	Path	**Required**. 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 914) to retrieve a list of all the security roles to determine the role's ID.

Table 386: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.2 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 Overview) are required to use this feature:
SecuritySettings: *Read*

Table 387: GET Security Roles {id} Global Permissions Input Parameters

Name	In	Description
id	Path	**Required**. 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 914) to retrieve a list of all the security roles to determine the role's ID.

Table 388: GET Security Roles {id} Global Permissions Response Data

Name	Description
	An object containing information about the global permissions granted to the security role. Details include: **Name** / **Description** Area — A string containing the name of the permission (e.g. "Certificates"). Permission — A string indicating the permission level granted in the area for this role (e.g. "Read").

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.3 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 displays a list of valid global permissions on the endpoint.

Tip: The following permissions (see Security Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 389: POST Security Roles {id}Global Permissions Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>id</td><td>Path</td><td>Required. 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 914) to retrieve a list of all the security roles to determine the role's ID.</td></tr>
<tr><td>glob-alPermissions</td><td>Body</td><td>An object containing information about the global permissions granted for this security role. Details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Area</td><td>Required. A string indicating the name of the permissions to grant (e.g. "AdminPortal").</td></tr>
<tr><td>Permis-sion</td><td>Required. A string indicating the permission level to grant (e.g. "Read"). Possible values are:
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>AdminPortal (a.k.a. Management Portal)</td><td>Read</td><td>Users can access the Management Portal. This permission must be enabled for all roles that will access the Management Portal.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Read</td><td>Users can view the agent auto-regis-tration settings; Users must also have Read permissions for Agent Management.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Modify</td><td>Users can modify the agent auto-regis-tration settings.</td></tr>
<tr><td>AgentManagement</td><td>Read</td><td>Users can access the Management Portal areas and API endpoints to:</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** • View orchestrators, including filtering the orchestrator management grid • View orchestrator jobs, including status, schedules, failures and warnings AgentManagement / Modify / Users can access the Management Portal areas and API endpoints to: • Manage orchestrators, including approving and disapproving them • Unschedule and reschedule orchestrator jobs

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** API / Read / Users can call the Classic (CMS) API endpoints. ApplicationSettings / Read / Users can view the application settings. ApplicationSettings / Modify / Users can modify the application settings. Auditing / Read / Users can access the Audit Log page in the Management Portal, and will be able to make API requests to obtain data from the audit log (query, etc.). The System Settings drop-down menu will display the Audit Log option to users with the *Auditing* Read permission. CertificateCollections / Modify / Users can add or edit certificate collections. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information. CertificateEnrollment / EnrollPFX / Users can use the PFX Enrollment page in the Management Portal and use the PFX enrollment related API endpoints.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** CertificateEnrollment / EnrollCSR / Users can use the CSR Enrollment page in the Management Portal and use the CSR enrollment related API endpoints. CertificateEnrollment / CsrGeneration / Users can use the CSR Generation page in the Management Portal and use the CSR generation related API endpoints. CertificateEnrollment / PendingCsr / Users can use manage pending CSRs. CertificateMetadataTypes / Read / Users can read custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints. CertificateMetadataTypes / Modify / Users can add, edit, and delete custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints. CertificateStoreManagement / Read / Users can view certificate stores—including the stores and containers but not

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** discovery records—and certificate store types. Users who also have Read permissions for *Certificates* can view inventory for a certificate store. See *Container Permissions* in the *Keyfactor Command Reference Guide* for more information. CertificateStoreManagement / Modify / Users can manage certificate stores—including the stores, containers, and discovery process—and certificate store types. Note that this permission does not control additions of certificates to certificate stores. CertificateStoreManagement / Schedule / Users can add certificates to certificate stores, renew/reissue certificates, and remove certificates from certificate stores. Certificates / Read / Users can view certificates in certificate search and certificate collections in the Management Portal

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** and with related API endpoints, including certificate history, and can download certificates. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information. Certificates / Import / Users can import certificates through Add Certificate in the Management Portal and with related API endpoints. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores from Add Certificate. Certificates / Recover / Users can download the certificates with their private key.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** Certificates / Revoke / Users can revoke certificates through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints. Certificates / Delete / Users can delete certificates and, if applicable, the private keys of the certificates from the Keyfactor Command database. Certificates / ImportPrivateKey / Users can save the private key for the certificate in the Keyfactor Command database. Certificates / EditMetadata / Users can modify certificate metadata for certificates accessed through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints. Dashboard / Read / Users can view the panels on their personalized dashboard and add and remove them. Dashboard / RiskHeader / Users can view the risk header at the top of

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** the dashboard. EventHandlerRegistration / Read / Users can view the event handler registration settings. EventHandlerRegistration / Modify / Users can modify the event handler registration settings. MacAutoEnrollManagement / Read / Users can view the Mac Auto-Enroll Management settings. MacAutoEnrollManagement / Modify / Users can modify the Mac Auto-Enroll Management settings. Monitoring / Read / Users can view the expiration alerts in the Certificate Alerts in the Management Portal and with related API endpoints, including the alert schedule. Monitoring / Modify / Users can modify the expiration alerts, including the alert text, recipients and event handlers. Users can also add new alerts, delete alerts and configure the expiration alert delivery schedule.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** Monitoring / Test / Users can test the expiration alerts, including sending email to recipients. Users must also have Read permissions for *Monitoring*. PkiManagement / Read / Users can view the Keyfactor Command PKI management settings within the following Management Portal areas and use related endpoints: • Certificate Authorities • Certificate Templates • Revocation Monitoring PkiManagement / Modify / Users can modify the Keyfactor Command PKI management settings: • Import, add, edit, and delete certificate authorities • Import certificate

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** templates • Add, edit, delete, and test revocation monitoring endpoints • Configure revocation monitoring schedule • Configure revocation monitoring recipients PrivilegedAccessManagement / Read / Users can view PAM providers. PrivilegedAccessManagement / Modify / Users can add, edit, and delete PAM providers. Reports / Read / Users can generate and view reports. Reports / Modify / Users can modify the delivery schedule for reports in Report Manager in the Management Portal and add, edit, and delete custom reports.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** **Note:** Report scheduling is limited by collection permissions. Users in roles that have *Reports: Read and Modify* permissions will also need to have *Read* collection permissions on individual collections to have the ability to add, edit and delete schedules associated with collections. The user will not have access to add, edit and delete schedules for any collections for which they do not have collection *Read* permissions in addition to *Reports*

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** permissions. SecuritySettings / Read / Users can view the settings for Security Roles and Security Identities. Users must also have the Read permission for *System Settings*. SecuritySettings / Modify / Users can modify the settings for Security Roles and Security Identities in the Management Portal and with related API endpoints. SSH / User / Users can generate their own SSH keys. SSH / ServerAdmin / Users can use all SSH functions, except creating server groups and assigning server group owners. Users have limited access to some functions based on server group ownership. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information. SSH / EnterpriseAdmin / Users can use all SSH

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** functions. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information. SslManagement / Read / Users can view the SSL Network Discovery and Monitoring area in the Management Portal and with related API endpoints, including defined networks and the network ranges configured for them, agent pools, and scan results. Users can use the query tool on the Results tab to find discovered endpoints and then view the discovered endpoints, including the details for the endpoints. SslManagement / Modify / Users can modify the SSL Network Discovery and Monitoring settings: • Create, edit, and delete networks, including scan

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** schedules and notification recipients • Add, edit, and delete network ranges for networks • Add, edit, and delete agent pools • Add and remove discovered endpoints from monitoring SystemSettings / Read / Users can view the System Settings for: • Application Settings • Event Handler Registration to view built-in or custom event handlers • API Applic-

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** ations allowed to use the APIs for certificate lifecycle manage-ment • SMTP Config-uration for email delivery of reports and alerts • Installed compon-ents • Licensing • Alerts and Warnings about the health of the Keyfactor Command system SystemSettings / Modify / Users can modify the System Settings for: • Applic-ation Settings to configure many options for

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** Keyfactor Command • Event Handler Registration to add or remove built-in or custom event handlers • Update SMTP Configuration for email delivery of reports and alerts • Installed components, including removing servers from use • Licensing, including the option to replace the existing license file WorkflowDefinitions / Read / Users can view the configured workflow

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** definitions. WorkflowDefinitions / Modify / Users can modify both the built-in and any custom workflow definitions, including the name and description and the configuration for the steps. Users can also add new workflow definitions, delete workflow definitions, publish workflow definitions, and import and export workflow definitions. WorkflowInstances / Manage / Users can manage initiated workflow instances, including stopping, restarting, and deleting them. WorkflowInstances / ReadAssignedToMe / Users can view the workflow instances that have been initiated and are awaiting input from them. **Tip:** There is not a security permission at this level that controls whether users can provide

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** input (a signal) to a workflow instance. This is controlled using the security roles configured on the specific workflow definition. Any user who holds one of the roles configured in the workflow step that requires a signal may provide the necessary input. The user does not need to hold the *ReadAssigned-ToMe Work-flowInstances* permission in order to provide the input. WorkflowInstances / ReadAll / Users can view all the workflow instances that have been initi-ated.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** WorkflowInstances / ReadMy / Users can view the workflow instances that have been initiated by them (e.g. because they enrolled for a certificate). WorkflowManagement (a.k.a. Alerts) / Read / Users can view the pending, issued, and denied workflow alerts. WorkflowManagement (a.k.a. Alerts) / Modify / Users can modify the pending, issued, and denied workflow alerts, including the alert text, recipients, and event handlers. Users can also add new alerts, delete alerts, and configure the pending alert delivery schedule. WorkflowManagement (a.k.a. Alerts) / Test / Users can test the pending alerts, including sending email to recipients. Users must also have Read permissions for *Workflow*. WorkflowManagement (a.k.a. Certificate Requests) / Participate / Users can participate in the pending, issued and denied workflow process by approving or denying certificate requests from the

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** Certificate Requests page or from the individual pages reached from links included in alerts in the Management Portal and with related API endpoints.

Table 390: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.4 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.

Warning: 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 391: PUT Security Roles {id}Global Permissions Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>id</td><td>Path</td><td>Required. 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 914) to retrieve a list of all the security roles to determine the role's ID.</td></tr>
<tr><td>glob-alPermissions</td><td>Body</td><td>An object containing information about the global permissions granted for this security role. Details include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Area</td><td>Required. A string indicating the name of the permissions to grant (e.g. "AdminPortal").</td></tr>
<tr><td>Permis-sion</td><td>Required. A string indicating the permission level to grant (e.g. "Read"). Possible values are:
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>AdminPortal (a.k.a. Management Portal)</td><td>Read</td><td>Users can access the Management Portal. This permission must be enabled for all roles that will access the Management Portal.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Read</td><td>Users can view the agent auto-regis-tration settings; Users must also have Read permissions for Agent Management.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Modify</td><td>Users can modify the agent auto-regis-tration settings.</td></tr>
<tr><td>AgentManagement</td><td>Read</td><td>Users can access the Management Portal areas and API endpoints to:</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** • View orchestrators, including filtering the orchestrator management grid • View orchestrator jobs, including status, schedules, failures and warnings
		AgentManagement / Modify / Users can access the Management Portal areas and API endpoints to: • Manage orchestrators, including approving and disapproving them • Unschedule and reschedule orchestrator jobs

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** API / Read / Users can call the Classic (CMS) API endpoints. ApplicationSettings / Read / Users can view the application settings. ApplicationSettings / Modify / Users can modify the application settings. Auditing / Read / Users can access the Audit Log page in the Management Portal, and will be able to make API requests to obtain data from the audit log (query, etc.). The System Settings drop-down menu will display the Audit Log option to users with the *Auditing* Read permission. CertificateCollections / Modify / Users can add or edit certificate collections. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information. CertificateEnrollment / EnrollPFX / Users can use the PFX Enrollment page in the Management Portal and use the PFX enrollment related API endpoints.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** CertificateEnrollment / EnrollCSR / Users can use the CSR Enrollment page in the Management Portal and use the CSR enrollment related API endpoints. CertificateEnrollment / CsrGeneration / Users can use the CSR Generation page in the Management Portal and use the CSR generation related API endpoints. CertificateEnrollment / PendingCsr / Users can use manage pending CSRs. CertificateMetadataTypes / Read / Users can read custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints. CertificateMetadataTypes / Modify / Users can add, edit, and delete custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints. CertificateStoreManagement / Read / Users can view certificate stores—including the stores and containers but not

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** discovery records—and certificate store types. Users who also have Read permissions for *Certificates* can view inventory for a certificate store. See *Container Permissions* in the *Keyfactor Command Reference Guide* for more information. CertificateStoreManagement / Modify / Users can manage certificate stores—including the stores, containers, and discovery process—and certificate store types. Note that this permission does not control additions of certificates to certificate stores. CertificateStoreManagement / Schedule / Users can add certificates to certificate stores, renew/reissue certificates, and remove certificates from certificate stores. Certificates / Read / Users can view certificates in certificate search and certificate collections in the Management Portal

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** and with related API endpoints, including certificate history, and can download certificates. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information. Certificates / Import / Users can import certificates through Add Certificate in the Management Portal and with related API endpoints. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores from Add Certificate. Certificates / Recover / Users can download the certificates with their private key.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** Certificates / Revoke / Users can revoke certificates through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints. Certificates / Delete / Users can delete certificates and, if applicable, the private keys of the certificates from the Keyfactor Command database. Certificates / ImportPrivateKey / Users can save the private key for the certificate in the Keyfactor Command database. Certificates / EditMetadata / Users can modify certificate metadata for certificates accessed through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints. Dashboard / Read / Users can view the panels on their personalized dashboard and add and remove them. Dashboard / RiskHeader / Users can view the risk header at the top of

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** the dashboard. EventHandlerRegistration / Read / Users can view the event handler registration settings. EventHandlerRegistration / Modify / Users can modify the event handler registration settings. MacAutoEnrollManagement / Read / Users can view the Mac Auto-Enroll Management settings. MacAutoEnrollManagement / Modify / Users can modify the Mac Auto-Enroll Management settings. Monitoring / Read / Users can view the expiration alerts in the Certificate Alerts in the Management Portal and with related API endpoints, including the alert schedule. Monitoring / Modify / Users can modify the expiration alerts, including the alert text, recipients and event handlers. Users can also add new alerts, delete alerts and configure the expiration alert delivery schedule.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** Monitoring / Test / Users can test the expiration alerts, including sending email to recipients. Users must also have Read permissions for *Monitoring*. PkiManagement / Read / Users can view the Keyfactor Command PKI management settings within the following Management Portal areas and use related endpoints: • Certificate Authorities • Certificate Templates • Revocation Monitoring PkiManagement / Modify / Users can modify the Keyfactor Command PKI management settings: • Import, add, edit, and delete certificate authorities • Import certificate

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** templates • Add, edit, delete, and test revocation monitoring endpoints • Configure revocation monitoring schedule • Configure revocation monitoring recipients PrivilegedAccessManagement / Read / Users can view PAM providers. PrivilegedAccessManagement / Modify / Users can add, edit, and delete PAM providers. Reports / Read / Users can generate and view reports. Reports / Modify / Users can modify the delivery schedule for reports in Report Manager in the Management Portal and add, edit, and delete custom reports.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** **Note:** Report scheduling is limited by collection permissions. Users in roles that have *Reports: Read and Modify* permissions will also need to have *Read* collection permissions on individual collections to have the ability to add, edit and delete schedules associated with collections. The user will not have access to add, edit and delete schedules for any collections for which they do not have collection *Read* permissions in addition to *Reports*

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** permissions. SecuritySettings / Read / Users can view the settings for Security Roles and Security Identities. Users must also have the Read permission for *System Settings*. SecuritySettings / Modify / Users can modify the settings for Security Roles and Security Identities in the Management Portal and with related API endpoints. SSH / User / Users can generate their own SSH keys. SSH / ServerAdmin / Users can use all SSH functions, except creating server groups and assigning server group owners. Users have limited access to some functions based on server group ownership. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information. SSH / EnterpriseAdmin / Users can use all SSH

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** functions. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information. SslManagement / Read / Users can view the SSL Network Discovery and Monitoring area in the Management Portal and with related API endpoints, including defined networks and the network ranges configured for them, agent pools, and scan results. Users can use the query tool on the Results tab to find discovered endpoints and then view the discovered endpoints, including the details for the endpoints. SslManagement / Modify / Users can modify the SSL Network Discovery and Monitoring settings: • Create, edit, and delete networks, including scan

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** schedules and notification recipients • Add, edit, and delete network ranges for networks • Add, edit, and delete agent pools • Add and remove discovered endpoints from monitoring SystemSettings / Read / Users can view the System Settings for: • Application Settings • Event Handler Registration to view built-in or custom event handlers • API Applic-

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** ations allowed to use the APIs for certificate lifecycle manage-ment • SMTP Config-uration for email delivery of reports and alerts • Installed compon-ents • Licensing • Alerts and Warnings about the health of the Keyfactor Command system SystemSettings / Modify / Users can modify the System Settings for: • Applic-ation Settings to configure many options for

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** Keyfactor Command • Event Handler Regis-tration to add or remove built-in or custom event handlers • Update SMTP Config-uration for email delivery of reports and alerts • Installed compon-ents, including removing servers from use • Licensing, including the option to replace the existing license file WorkflowDefinitions / Read / Users can view the configured workflow

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** definitions. WorkflowDefinitions / Modify / Users can modify both the built-in and any custom workflow definitions, including the name and description and the configuration for the steps. Users can also add new workflow definitions, delete workflow definitions, publish workflow definitions, and import and export workflow definitions. WorkflowInstances / Manage / Users can manage initiated workflow instances, including stopping, restarting, and deleting them. WorkflowInstances / ReadAssignedToMe / Users can view the workflow instances that have been initiated and are awaiting input from them. **Tip:** There is not a security permission at this level that controls whether users can provide

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** input (a signal) to a workflow instance. This is controlled using the security roles configured on the specific workflow definition. Any user who holds one of the roles configured in the workflow step that requires a signal may provide the necessary input. The user does not need to hold the *ReadAssigned-ToMe Work-flowInstances* permission in order to provide the input. WorkflowInstances / ReadAll / Users can view all the workflow instances that have been initi-ated.

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** WorkflowInstances / ReadMy / Users can view the workflow instances that have been initiated by them (e.g. because they enrolled for a certificate). WorkflowManagement (a.k.a. Alerts) / Read / Users can view the pending, issued, and denied workflow alerts. WorkflowManagement (a.k.a. Alerts) / Modify / Users can modify the pending, issued, and denied workflow alerts, including the alert text, recipients, and event handlers. Users can also add new alerts, delete alerts, and configure the pending alert delivery schedule. WorkflowManagement (a.k.a. Alerts) / Test / Users can test the pending alerts, including sending email to recipients. Users must also have Read permissions for *Workflow*. WorkflowManagement (a.k.a. Certificate Requests) / Participate / Users can participate in the pending, issued and denied workflow process by approving or denying certificate requests from the

Name	In	Description
		Name / **Description** **Name** / **Value** / **Description** Certificate Requests page or from the individual pages reached from links included in alerts in the Management Portal and with related API endpoints.

Table 392: 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** / **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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.5 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 Overview) are required to use this feature:
SecuritySettings: *Read*

Table 393: GET Security Roles {id} Permissions Containers Input Parameters

Name	In	Description
id	Path	**Required**. 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 914) to retrieve a list of all the security roles to determine the role's ID.

Table 394: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.6 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 395: POST Security Roles {id} Permissions Containers Input Parameters

Name	In	Description
id	Path	**Required**. 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 914) 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 396: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.7 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.

Warning: 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 397: PUT Security Roles {id} Permissions Containers Input Parameters

Name	In	Description
id	Path	**Required**. 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 914) 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 398: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.8 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 Overview) are required to use this feature:
SecuritySettings: *Read*

Table 399: GET Security Roles {id} Permissions Collections Input Parameters

Name	In	Description
id	Path	**Required**. 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 914) to retrieve a list of all the security roles to determine the role's ID.

Table 400: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.9 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 401: POST Security Roles {id} Permissions Collections Input Parameters

Name	In	Description
id	Path	**Required**. 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 914) 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 402: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.23.10 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.

Warning: 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 403: PUT Security Roles {id} Permissions Collections Input Parameters

Name	In	Description
id	Path	**Required**. 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 914) 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 404: 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 — **Required**. An integer containing the collection ID. Name — **Required**. A string containing the name of the certificate collection . Permission — **Required**. A string indicating the permission granted on the entity for this role.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.24 Security Roles

The Security Roles component of the Keyfactor API includes methods necessary to list, add, update, and delete security roles. The permissions set with these methods are used to control access to all aspects of Keyfactor Command.

Table 405: Security Roles Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes the security role with the specified ID.	DELETE Security Roles ID below
/{id}	GET	Returns details for the security role with the specified ID, including permissions granted to the role and security identities assigned the role.	GET Security Roles ID on the next page
/{id}/Identities	GET	Returns the security identities assigned to the security role with the specified ID.	GET Security Roles ID Identities on page 912
/{id}/Identities	PUT	Updates the security identities assigned to the security role with the specified ID.	PUT Security Roles ID Identities on page 913
/	GET	Returns all security roles with filtering and output options.	GET Security Roles on page 914
/	POST	Adds a new security role.	POST Security Roles on page 916
/	PUT	Updates the security role with the specified ID.	PUT Security Roles on page 933
/{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 950

2.2.24.1 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 914) 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 406: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.24.2 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 Overview) are required to use this feature:
SecuritySettings: *Read*

Table 407: GET Security Roles {id} 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 914) to retrieve a list of all the security roles to determine the role's ID.

Table 408: GET Security Roles {id} 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 (true) or not (false). 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.
Identities	An array 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.
SID	A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.

Permissions	An object containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. For example: `"Permissions": [`

Name	Description
	<pre>"AdminPortal:Read", "Dashboard:Read"],</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.24.3 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 Overview) are required to use this feature:
SecuritySettings: *Read*

Table 409: 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 914) to retrieve a list of all the security roles to determine the role's ID.

Table 410: GET Security Roles {id} Identities Response Data

Name	Description
	An array 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. 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.24.4 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 411: 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 the next page) 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 835) to retrieve a list of all the security identities to determine the identity ID(s).

Table 412: PUT Security Roles {id} Identities Response Data

Name	Description
	An array 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.
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.24.5 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 Overview) are required to use this feature:
SecuritySettings: *Read*

Table 413: GET Security Roles 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Security Role Search Feature*. 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 414: GET Security Roles 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 (true) or not (false). 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.
Identities	An array 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. SID — A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.
Permissions	An object containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. For example: `"Permissions": [`

Name	Description
	<pre>"AdminPortal:Read", "Dashboard:Read"],</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.24.6 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 415: POST Security Roles 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>Permissions</td><td>Body</td><td>An object containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. Possible values are:
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>AdminPortal
(a.k.a. Management Portal)</td><td>Read</td><td>Users can access the Management Portal. This permission must be enabled for all roles that will access the Management Portal.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Read</td><td>Users can view the agent auto-registration settings; Users must also have Read permissions for Agent Management.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Modify</td><td>Users can modify the agent auto-registration settings.</td></tr>
<tr><td>AgentManagement</td><td>Read</td><td>Users can access the Management Portal areas and API endpoints to:
View orchestrators, including filtering the orchestrator management grid</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>• View orchestrator jobs, including status, schedules, failures and warnings</td></tr><tr><td>AgentManagement</td><td>Modify</td><td>Users can access the Management Portal areas and API endpoints to: • Manage orchestrators, including approving and disapproving them • Unschedule and reschedule orchestrator jobs</td></tr><tr><td>API</td><td>Read</td><td>Users can call the Classic (CMS) API endpoints.</td></tr><tr><td>ApplicationSettings</td><td>Read</td><td>Users can view the application settings.</td></tr><tr><td>ApplicationSettings</td><td>Modify</td><td>Users can modify the application settings.</td></tr><tr><td>Auditing</td><td>Read</td><td>Users can access the Audit Log page in the Management Portal, and will be able to make API requests to obtain data from the audit log (query, etc.). The System Settings drop-down menu will display the Audit Log option to users with the Auditing Read permission.</td></tr><tr><td>CertificateCollections</td><td>Modify</td><td>Users can add or edit certi-</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>ficate collections. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information.</td></tr><tr><td>CertificateEnrollment</td><td>EnrollPFX</td><td>Users can use the PFX Enrollment page in the Management Portal and use the PFX enrollment related API endpoints.</td></tr><tr><td>CertificateEnrollment</td><td>EnrollCSR</td><td>Users can use the CSR Enrollment page in the Management Portal and use the CSR enrollment related API endpoints.</td></tr><tr><td>CertificateEnrollment</td><td>CsrGeneration</td><td>Users can use the CSR Generation page in the Management Portal and use the CSR generation related API endpoints.</td></tr><tr><td>CertificateEnrollment</td><td>PendingCsr</td><td>Users can use manage pending CSRs.</td></tr><tr><td>CertificateMetadataTypes</td><td>Read</td><td>Users can read custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints.</td></tr><tr><td>CertificateMetadataTypes</td><td>Modify</td><td>Users can add, edit, and delete custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints.</td></tr><tr><td>CertificateStoreManagement</td><td>Read</td><td>Users can view certificate</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>stores—including the stores and containers but not discovery records—and certificate store types. Users who also have Read permissions for *Certificates* can view inventory for a certificate store. See *Container Permissions* in the *Keyfactor Command Reference Guide* for more information.</td></tr><tr><td>CertificateStoreManagement</td><td>Modify</td><td>Users can manage certificate stores—including the stores, containers, and discovery process—and certificate store types. Note that this permission does not control additions of certificates to certificate stores.</td></tr><tr><td>CertificateStoreManagement</td><td>Schedule</td><td>Users can add certificates to certificate stores, renew/reissue certificates, and remove certificates from certificate stores.</td></tr><tr><td>Certificates</td><td>Read</td><td>Users can view certificates in certificate search and certificate collections in the Management Portal and with related API endpoints, including certificate history, and can download certificates. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores.</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information.</td></tr><tr><td>Certificates</td><td>Import</td><td>Users can import certificates through Add Certificate in the Management Portal and with related API endpoints. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores from Add Certificate.</td></tr><tr><td>Certificates</td><td>Recover</td><td>Users can download the certificates with their private key.</td></tr><tr><td>Certificates</td><td>Revoke</td><td>Users can revoke certificates through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints.</td></tr><tr><td>Certificates</td><td>Delete</td><td>Users can delete certificates and, if applicable, the private keys of the certificates from the Keyfactor Command database.</td></tr><tr><td>Certificates</td><td>ImportPrivateKey</td><td>Users can save the private key for the certificate in the Keyfactor Command database.</td></tr><tr><td>Certificates</td><td>EditMetadata</td><td>Users can modify certificate metadata for certificates accessed through Certificate Search and Certificate Collections in the Management</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>Portal and with related API endpoints.</td></tr><tr><td>Dashboard</td><td>Read</td><td>Users can view the panels on their personalized dashboard and add and remove them.</td></tr><tr><td>Dashboard</td><td>RiskHeader</td><td>Users can view the risk header at the top of the dashboard.</td></tr><tr><td>EventHandlerRegistration</td><td>Read</td><td>Users can view the event handler registration settings.</td></tr><tr><td>EventHandlerRegistration</td><td>Modify</td><td>Users can modify the event handler registration settings.</td></tr><tr><td>MacAutoEnrollManagement</td><td>Read</td><td>Users can view the Mac Auto-Enroll Management settings.</td></tr><tr><td>MacAutoEnrollManagement</td><td>Modify</td><td>Users can modify the Mac Auto-Enroll Management settings.</td></tr><tr><td>Monitoring</td><td>Read</td><td>Users can view the expiration alerts in the Certificate Alerts in the Management Portal and with related API endpoints, including the alert schedule.</td></tr><tr><td>Monitoring</td><td>Modify</td><td>Users can modify the expiration alerts, including the alert text, recipients and event handlers. Users can also add new alerts, delete alerts and configure the expiration alert delivery schedule.</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>Value</th><th>Description</th></tr>
<tr><td>Monitoring</td><td>Test</td><td>Users can test the expiration alerts, including sending email to recipients. Users must also have Read permissions for Monitoring.</td></tr>
<tr><td>PkiManagement</td><td>Read</td><td>Users can view the Keyfactor Command PKI management settings within the following Management Portal areas and use related endpoints:
Certificate Authorities
Certificate Templates
Revocation Monitoring</td></tr>
<tr><td>PkiManagement</td><td>Modify</td><td>Users can modify the Keyfactor Command PKI management settings:
Import, add, edit, and delete certificate authorities
Import certificate templates
Add, edit, delete, and test revocation monitoring endpoints
Configure revocation monitoring schedule
Configure revocation monitoring recipients</td></tr>
<tr><td>PrivilegedAccessManagement</td><td>Read</td><td>Users can view PAM</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>Value</th><th>Description</th></tr>
<tr><td></td><td></td><td>providers.</td></tr>
<tr><td>PrivilegedAccessManagement</td><td>Modify</td><td>Users can add, edit, and delete PAM providers.</td></tr>
<tr><td>Reports</td><td>Read</td><td>Users can generate and view reports.</td></tr>
<tr><td>Reports</td><td>Modify</td><td>Users can modify the delivery schedule for reports in Report Manager in the Management Portal and add, edit, and delete custom reports.

Note: Report scheduling is limited by collection permissions. Users in roles that have Reports: Read and Modify permissions will also need to have Read collection permissions on individual collections to have the ability to add, edit and delete schedules associated with collections. The user will not have access to add, edit and delete schedules for any collections for which they do not have collection Read permissions in addition to Reports permissions.</td></tr>
</table>
</td></tr>
</table>

Name	In	Description

Name	Value	Description
SecuritySettings	Read	Users can view the settings for Security Roles and Security Identities. Users must also have the Read permission for *System Settings*.
SecuritySettings	Modify	Users can modify the settings for Security Roles and Security Identities in the Management Portal and with related API endpoints.
SSH	User	Users can generate their own SSH keys.
SSH	ServerAdmin	Users can use all SSH functions, except creating server groups and assigning server group owners. Users have limited access to some functions based on server group ownership. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information.
SSH	EnterpriseAdmin	Users can use all SSH functions. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information.
SslManagement	Read	Users can view the SSL Network Discovery and Monitoring area in the Management Portal and with related API endpoints, including defined networks

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>and the network ranges configured for them, agent pools, and scan results. Users can use the query tool on the Results tab to find discovered endpoints and then view the discovered endpoints, including the details for the endpoints.</td></tr><tr><td>SslManagement</td><td>Modify</td><td>Users can modify the SSL Network Discovery and Monitoring settings: - Create, edit, and delete networks, including scan schedules and notification recipients - Add, edit, and delete network ranges for networks - Add, edit, and delete agent pools - Add and remove discovered endpoints from monitoring </td></tr><tr><td>SystemSettings</td><td>Read</td><td>Users can view the System Settings for: - Application Settings - Event Handler Registration to view built-in or custom event handlers </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>Value</th><th>Description</th></tr>
<tr><td></td><td></td><td>• API Applications allowed to use the APIs for certificate lifecycle management
• SMTP Configuration for email delivery of reports and alerts
• Installed components
• Licensing
• Alerts and Warnings about the health of the Keyfactor Command system</td></tr>
<tr><td>SystemSettings</td><td>Modify</td><td>Users can modify the System Settings for:
• Application Settings to configure many options for Keyfactor Command
• Event Handler Registration to add or remove built-in or custom event handlers
• Update SMTP Configuration for email delivery of reports and alerts</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td> - Installed components, including removing servers from use - Licensing, including the option to replace the existing license file </td></tr><tr><td>WorkflowDefinitions</td><td>Read</td><td>Users can view the configured workflow definitions.</td></tr><tr><td>WorkflowDefinitions</td><td>Modify</td><td>Users can modify both the built-in and any custom workflow definitions, including the name and description and the configuration for the steps. Users can also add new workflow definitions, delete workflow definitions, publish workflow definitions, and import and export workflow definitions.</td></tr><tr><td>WorkflowInstances</td><td>Manage</td><td>Users can manage initiated workflow instances, including stopping, restarting, and deleting them.</td></tr><tr><td>WorkflowInstances</td><td>ReadAssignedToMe</td><td>Users can view the workflow instances that have been initiated and are awaiting input from them.  Tip: There is not a</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>security permission at this level that controls whether users can provide input (a signal) to a workflow instance. This is controlled using the security roles configured on the specific workflow definition. Any user who holds one of the roles configured in the workflow step that requires a signal may provide the necessary input. The user does not need to hold the *ReadAssignedToMeWorkflowInstances* permission in order to provide the input.</td></tr><tr><td>WorkflowInstances</td><td>ReadAll</td><td>Users can view all the workflow instances that have been initiated.</td></tr><tr><td>WorkflowInstances</td><td>ReadMy</td><td>Users can view the workflow instances that have been initiated by them (e.g. because they enrolled for a certificate).</td></tr><tr><td>WorkflowManagement (a.k.a. Alerts)</td><td>Read</td><td>Users can view the pending, issued, and denied workflow alerts.</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>Value</th><th>Description</th></tr>
<tr><td>WorkflowManagement (a.k.a. Alerts)</td><td>Modify</td><td>Users can modify the pending, issued, and denied workflow alerts, including the alert text, recipients, and event handlers. Users can also add new alerts, delete alerts, and configure the pending alert delivery schedule.</td></tr>
<tr><td>WorkflowManagement (a.k.a. Alerts)</td><td>Test</td><td>Users can test the pending alerts, including sending email to recipients. Users must also have Read permissions for Workflow.</td></tr>
<tr><td>WorkflowManagement (a.k.a. Certificate Requests)</td><td>Participate</td><td>Users can participate in the pending, issued and denied workflow process by approving or denying certificate requests from the Certificate Requests page or from the individual pages reached from links included in alerts in the Management Portal and with related API endpoints.</td></tr>
</table>
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre>
</td></tr>
<tr><td>Identities</td><td>Body</td><td>An array containing one or more identifiers for each security identity to associate with the role. Supported identifiers include:</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td>AccountName</td><td>**Required***. A string containing the account name for the security identity. For Active Directory user and groups, this will be in the form DOMAIN\\user or group name. For example: KEYEXAMPLE\\PKI Administrators * One of *AccountName* or *SID* is required in 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": [{ "Name": "KEYEXAMPLE\\jsmith" }, { "Name": "KEYEXAMPLE\\mjones" }]</pre>

Table 416: POST Security Roles 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 (true) or not (false). 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.
Identities	An array 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. SID — A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.
Permissions	An object containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. For example: `"Permissions": [`

Name	Description
	<pre>"AdminPortal:Read", "Dashboard:Read"],</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.24.7 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Warning: 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 417: PUT Security Roles 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 GET Security Roles on page 914) 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>Permissions</td><td>Body</td><td>An object containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. Possible values are:
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>AdminPortal
(a.k.a. Management Portal)</td><td>Read</td><td>Users can access the Management Portal. This permission must be enabled for all roles that will access the Management Portal.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Read</td><td>Users can view the agent auto-registration settings; Users must also have Read permissions for Agent Management.</td></tr>
<tr><td>AgentAutoRegistration</td><td>Modify</td><td>Users can modify the agent auto-registration settings.</td></tr>
<tr><td>AgentManagement</td><td>Read</td><td>Users can access the Management Portal areas and API endpoints to:
View orches-trators, including</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>filtering the orchestrator management grid • View orchestrator jobs, including status, schedules, failures and warnings</td></tr><tr><td>AgentManagement</td><td>Modify</td><td>Users can access the Management Portal areas and API endpoints to: • Manage orchestrators, including approving and disapproving them • Unschedule and reschedule orchestrator jobs</td></tr><tr><td>API</td><td>Read</td><td>Users can call the Classic (CMS) API endpoints.</td></tr><tr><td>ApplicationSettings</td><td>Read</td><td>Users can view the application settings.</td></tr><tr><td>ApplicationSettings</td><td>Modify</td><td>Users can modify the application settings.</td></tr><tr><td>Auditing</td><td>Read</td><td>Users can access the Audit Log page in the Management Portal, and will be able to make API requests to obtain data from the audit log (query, etc.). The System Settings drop-down menu</td></tr></table>

Name	In	Description

Name	Value	Description
		will display the Audit Log option to users with the *Auditing* Read permission.
CertificateCollections	Modify	Users can add or edit certificate collections. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information.
CertificateEnrollment	EnrollPFX	Users can use the PFX Enrollment page in the Management Portal and use the PFX enrollment related API endpoints.
CertificateEnrollment	EnrollCSR	Users can use the CSR Enrollment page in the Management Portal and use the CSR enrollment related API endpoints.
CertificateEnrollment	CsrGeneration	Users can use the CSR Generation page in the Management Portal and use the CSR generation related API endpoints.
CertificateEnrollment	PendingCsr	Users can use manage pending CSRs.
CertificateMetadataTypes	Read	Users can read custom metadata attribute definitions on the Certificate Metadata page in the Management Portal and with related API endpoints.
CertificateMetadataTypes	Modify	Users can add, edit, and delete custom metadata attribute definitions on the

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>Certificate Metadata page in the Management Portal and with related API endpoints.</td></tr><tr><td>CertificateStoreManagement</td><td>Read</td><td>Users can view certificate stores—including the stores and containers but not discovery records—and certificate store types. Users who also have Read permissions for Certificates can view inventory for a certificate store. See Container Permissions in the Keyfactor Command Reference Guide for more information.</td></tr><tr><td>CertificateStoreManagement</td><td>Modify</td><td>Users can manage certificate stores—including the stores, containers, and discovery process—and certificate store types. Note that this permission does not control additions of certificates to certificate stores.</td></tr><tr><td>CertificateStoreManagement</td><td>Schedule</td><td>Users can add certificates to certificate stores, renew/reissue certificates, and remove certificates from certificate stores.</td></tr><tr><td>Certificates</td><td>Read</td><td>Users can view certificates in certificate search and certificate collections in the Management Portal and with related API endpoints, including certificate history, and can download certificates. Users who also have</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores. See *Certificate Permissions* in the *Keyfactor Command Reference Guide* for more information.</td></tr><tr><td>Certificates</td><td>Import</td><td>Users can import certificates through Add Certificate in the Management Portal and with related API endpoints. Users who also have Read permissions for *Certificate Store Management* or container permissions can add certificates to certificate stores from Add Certificate.</td></tr><tr><td>Certificates</td><td>Recover</td><td>Users can download the certificates with their private key.</td></tr><tr><td>Certificates</td><td>Revoke</td><td>Users can revoke certificates through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints.</td></tr><tr><td>Certificates</td><td>Delete</td><td>Users can delete certificates and, if applicable, the private keys of the certificates from the Keyfactor Command database.</td></tr><tr><td>Certificates</td><td>ImportPrivateKey</td><td>Users can save the private key for the certificate in the Keyfactor Command database.</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>Certificates</td><td>EditMetadata</td><td>Users can modify certificate metadata for certificates accessed through Certificate Search and Certificate Collections in the Management Portal and with related API endpoints.</td></tr><tr><td>Dashboard</td><td>Read</td><td>Users can view the panels on their personalized dashboard and add and remove them.</td></tr><tr><td>Dashboard</td><td>RiskHeader</td><td>Users can view the risk header at the top of the dashboard.</td></tr><tr><td>EventHandlerRegistration</td><td>Read</td><td>Users can view the event handler registration settings.</td></tr><tr><td>EventHandlerRegistration</td><td>Modify</td><td>Users can modify the event handler registration settings.</td></tr><tr><td>MacAutoEnrollManagement</td><td>Read</td><td>Users can view the Mac Auto-Enroll Management settings.</td></tr><tr><td>MacAutoEnrollManagement</td><td>Modify</td><td>Users can modify the Mac Auto-Enroll Management settings.</td></tr><tr><td>Monitoring</td><td>Read</td><td>Users can view the expiration alerts in the Certificate Alerts in the Management Portal and with related API endpoints, including the alert schedule.</td></tr><tr><td>Monitoring</td><td>Modify</td><td>Users can modify the expiration alerts, including the alert text, recipients and event handlers. Users can</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>Value</th><th>Description</th></tr>
<tr><td></td><td></td><td>also add new alerts, delete alerts and configure the expiration alert delivery schedule.</td></tr>
<tr><td>Monitoring</td><td>Test</td><td>Users can test the expiration alerts, including sending email to recipients. Users must also have Read permissions for Monitoring.</td></tr>
<tr><td>PkiManagement</td><td>Read</td><td>Users can view the Keyfactor Command PKI management settings within the following Management Portal areas and use related endpoints:
Certificate Authorities
Certificate Templates
Revocation Monitoring</td></tr>
<tr><td>PkiManagement</td><td>Modify</td><td>Users can modify the Keyfactor Command PKI management settings:
Import, add, edit, and delete certificate authorities
Import certificate templates
Add, edit, delete, and test revocation monitoring endpoints
Configure revocation monitoring schedule</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>• Configure revocation monitoring recipients</td></tr><tr><td>PrivilegedAccessManagement</td><td>Read</td><td>Users can view PAM providers.</td></tr><tr><td>PrivilegedAccessManagement</td><td>Modify</td><td>Users can add, edit, and delete PAM providers.</td></tr><tr><td>Reports</td><td>Read</td><td>Users can generate and view reports.</td></tr><tr><td>Reports</td><td>Modify</td><td>Users can modify the delivery schedule for reports in Report Manager in the Management Portal and add, edit, and delete custom reports. **Note:** Report scheduling is limited by collection permissions. Users in roles that have *Reports: Read and Modify* permissions will also need to have *Read* collection permissions on individual collections to have the ability to add, edit and delete schedules associated with collections. The user will not have access to add, edit and delete schedules for any collections for which they do not</td></tr></table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>have collection *Read* permissions in addition to *Reports* permissions.</td></tr><tr><td>SecuritySettings</td><td>Read</td><td>Users can view the settings for Security Roles and Security Identities. Users must also have the Read permission for *System Settings*.</td></tr><tr><td>SecuritySettings</td><td>Modify</td><td>Users can modify the settings for Security Roles and Security Identities in the Management Portal and with related API endpoints.</td></tr><tr><td>SSH</td><td>User</td><td>Users can generate their own SSH keys.</td></tr><tr><td>SSH</td><td>ServerAdmin</td><td>Users can use all SSH functions, except creating server groups and assigning server group owners. Users have limited access to some functions based on server group ownership. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information.</td></tr><tr><td>SSH</td><td>EnterpriseAdmin</td><td>Users can use all SSH functions. See *SSH Permissions* in the *Keyfactor Command Reference Guide* for more information.</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>Value</th><th>Description</th></tr>
<tr><td>SslManagement</td><td>Read</td><td>Users can view the SSL Network Discovery and Monitoring area in the Management Portal and with related API endpoints, including defined networks and the network ranges configured for them, agent pools, and scan results. Users can use the query tool on the Results tab to find discovered endpoints and then view the discovered endpoints, including the details for the endpoints.</td></tr>
<tr><td>SslManagement</td><td>Modify</td><td>Users can modify the SSL Network Discovery and Monitoring settings:
Create, edit, and delete networks, including scan schedules and notification recipients
Add, edit, and delete network ranges for networks
Add, edit, and delete agent pools
Add and remove discovered endpoints from monitoring</td></tr>
<tr><td>SystemSettings</td><td>Read</td><td>Users can view the System Settings for:
Application Settings</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>Value</th><th>Description</th></tr>
<tr><td></td><td></td><td>• Event Handler Registration to view built-in or custom event handlers
• API Applications allowed to use the APIs for certificate lifecycle management
• SMTP Configuration for email delivery of reports and alerts
• Installed components
• Licensing
• Alerts and Warnings about the health of the Keyfactor Command system</td></tr>
<tr><td>SystemSettings</td><td>Modify</td><td>Users can modify the System Settings for:
• Application Settings to configure many options for Keyfactor Command
• Event Handler Registration to add or remove built-in or custom event handlers
• Update SMTP</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td></td><td></td><td>Configuration for email delivery of reports and alerts • Installed components, including removing servers from use • Licensing, including the option to replace the existing license file</td></tr><tr><td>WorkflowDefinitions</td><td>Read</td><td>Users can view the configured workflow definitions.</td></tr><tr><td>WorkflowDefinitions</td><td>Modify</td><td>Users can modify both the built-in and any custom workflow definitions, including the name and description and the configuration for the steps. Users can also add new workflow definitions, delete workflow definitions, publish workflow definitions, and import and export workflow definitions.</td></tr><tr><td>WorkflowInstances</td><td>Manage</td><td>Users can manage initiated workflow instances, including stopping, restarting, and deleting them.</td></tr><tr><td>WorkflowInstances</td><td>ReadAssignedToMe</td><td>Users can view the workflow instances that have been</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>Value</th><th>Description</th></tr>
<tr><td></td><td></td><td>initiated and are awaiting input from them.

Tip: There is not a security permission at this level that controls whether users can provide input (a signal) to a workflow instance. This is controlled using the security roles configured on the specific workflow definition. Any user who holds one of the roles configured in the workflow step that requires a signal may provide the necessary input. The user does not need to hold the ReadAssignedToMeWorkflowInstances permission in order to provide the input.</td></tr>
<tr><td>WorkflowInstances</td><td>ReadAll</td><td>Users can view all the workflow instances that have been initiated.</td></tr>
<tr><td>WorkflowInstances</td><td>ReadMy</td><td>Users can view the workflow instances that have been initiated by them (e.g. because they enrolled for a certificate).</td></tr>
<tr><td>WorkflowManagement</td><td>Read</td><td>Users can view the pending,</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>Value</th><th>Description</th></tr>
<tr><td>(a.k.a. Alerts)</td><td></td><td>issued, and denied workflow alerts.</td></tr>
<tr><td>WorkflowManagement (a.k.a. Alerts)</td><td>Modify</td><td>Users can modify the pending, issued, and denied workflow alerts, including the alert text, recipients, and event handlers. Users can also add new alerts, delete alerts, and configure the pending alert delivery schedule.</td></tr>
<tr><td>WorkflowManagement (a.k.a. Alerts)</td><td>Test</td><td>Users can test the pending alerts, including sending email to recipients. Users must also have Read permissions for Workflow.</td></tr>
<tr><td>WorkflowManagement (a.k.a. Certificate Requests)</td><td>Participate</td><td>Users can participate in the pending, issued and denied workflow process by approving or denying certificate requests from the Certificate Requests page or from the individual pages reached from links included in alerts in the Management Portal and with related API endpoints.</td></tr>
</table>
For example:
<pre>"Permissions": [
   "AdminPortal:Read",
   "Dashboard:Read"
],</pre>
</td></tr>
<tr><td>Identities</td><td>Body</td><td>An array containing one or more identifiers for each security identity to associate with the role. Supported identifiers include:</td></tr>
</table>

Name	In	Description
		<table><tr><th>Name</th><th>Description</th></tr><tr><td>AccountName</td><td>**Required***. A string containing the account name for the security identity. For Active Directory user and groups, this will be in the form DOMAIN\\user or group name. For example: KEYEXAMPLE\\PKI Administrators * One of *AccountName* or *SID* is required in 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": [{ "Name": "KEYEXAMPLE\\jsmith" }, { "Name": "KEYEXAMPLE\\mjones" }]</pre>

Table 418: PUT Security Roles 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 (true) or not (false). 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.
Identities	An array 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. SID — A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.
Permissions	An object containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. For example: `"Permissions": [`

Name	Description
	<pre>"AdminPortal:Read", "Dashboard:Read"],</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.24.8 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 Overview) are required to use this feature:
SecuritySettings: *Modify*

Table 419: 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 914) 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** / **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.

Table 420: 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 (true) or not (false). 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.
Identities	An array 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. SID — A string containing the security identifier from the source identity store (e.g. Active Directory) for the security identity.
Permissions	An object containing the permissions assigned to the role in a comma-separated list of Name:Value pairs. For example: `"Permissions": [`

Name	Description
	<pre>"AdminPortal:Read", "Dashboard:Read"],</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.25 SSH

The SSH component of the Keyfactor Web APIs includes methods necessary to create, update, and delete SSH keys, logons, servers, server groups, and service accounts within Keyfactor Command.

Table 421: 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 956
/Keys/Unmanaged/{id}	GET	Retrieve details for a discovered unmanaged SSH key for the specified ID.	GET SSH Keys Unmanaged ID on page 957
/Keys/MyKey	GET	Retrieve details for a user's SSH key generated through Keyfactor Command.	GET SSH Keys My Key on page 958
/Keys/MyKey	POST	Generate a new SSH key pair for a user through Keyfactor Command.	POST SSH Keys My Key on page 961
/Keys/MyKey	PUT	Update an SSH key for a user through Keyfactor Command.	PUT SSH Keys My Key on page 965
/Keys/Unmanaged	DELETE	Delete one or more discovered unmanaged SSH keys based on a selection query.	DELETE SSH Keys Unmanaged on page 966
/Keys/Unmanaged	GET	Retrieve details for one or more discovered unmanaged SSH keys based on a selection query.	GET SSH Keys Unmanaged on page 967

Endpoint	Method	Description	Link
/Logons/{id}	DELETE	Deletes a Linux logon from Keyfactor Command.	DELETE SSH Logons ID on page 970
/Logons/{id}	GET	Returns information about a Linux logons.	GET SSH Logons ID on page 971
/Logons/	GET	Returns information about one or more Linux logons.	GET SSH Logons on page 973
/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 974
/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 977
/Servers/{id}	DELETE	Deletes the SSH server with the specified ID.	DELETE SSH Servers ID on page 979
/Servers/{id}	GET	Returns the SSH server with the specified ID.	GET SSH Servers ID on page 979
/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 984
/Servers/	GET	Returns a list of a SSH servers configured in Keyfactor Command.	GET SSH Servers on page 986
/Servers/	POST	Creates a new SSH server.	POST SSH Servers on page 990
/Servers/	PUT	Updates an existing SSH server.	PUT SSH Servers on page 995
/Servers/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server.	DELETE SSH Servers Access on page 1000
/Servers/Access	POST	Creates Linux logon to user and service account mappings for an SSH server.	POST SSH Servers Access on page 1002

Endpoint	Method	Description	Link
/ServerGroups/{id}	DELETE	Deletes the SSH server group with the specified ID.	DELETE SSH Server Groups ID on page 1005
/ServerGroups/{id}	GET	Returns the SSH server group with the specified ID.	GET SSH Server Groups ID on page 1006
/ServerGroups/{name}	GET	Returns the SSH server group with the specified name.	GET SSH Server Groups Name on page 1010
/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 1014
/ServerGroups/	GET	Returns a list of a SSH server groups configured in Keyfactor Command.	GET SSH Server Groups on page 1015
/ServerGroups/	POST	Creates a new SSH server group.	POST SSH Server Groups on page 1020
/ServerGroups/	PUT	Updates an existing SSH server group.	PUT SSH Server Groups on page 1027
/ServerGroups/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server group.	DELETE SSH Server Groups Access on page 1034
/ServerGroups/Access	POST	Creates Linux logon to user and service account mappings for an SSH server group.	POST SSH Server Groups Access on page 1035
/ServiceAccounts/{id}	DELETE	Deletes the SSH service account with the specified ID.	DELETE SSH Service Accounts ID on page 1038
/ServiceAccounts/{id}	GET	Returns the SSH service account with the specified ID.	GET SSH Service Accounts ID on page 1040
/ServiceAccounts/Key/{id}	GET	Returns the public key and optional private key	GET SSH Service

Endpoint	Method	Description	Link
		of an SSH service account with the specified ID.	Accounts Key ID on page 1046
/ServiceAccounts/	DELETE	Deletes one or more SSH service accounts with the specified IDs.	DELETE SSH Service Accounts on page 1050
/ServiceAccounts/	GET	Returns a list of SSH service accounts based on the specified filters.	GET SSH Service Accounts on page 1052
/ServiceAccounts/	POST	Creates a new SSH service account.	POST SSH Service Accounts on page 1059
/ServiceAccounts/	PUT	Updates an existing SSH service account.	PUT SSH Service Accounts on page 1068
/ServiceAccounts/Rotate/{id}	POST	Generates a new key pair for an existing service account.	POST SSH Service Accounts Rotate ID on page 1075
/Users/{id}	DELETE	Deletes the SSH user with the specified ID.	DELETE SSH Users ID on page 1079
/Users/{id}	GET	Returns the SSH user with the specified ID.	GET SSH Users ID on page 1079
/Users/	GET	Returns a list of SSH users based on the specified filters.	GET SSH Users on page 1084
/Users/	POST	Creates a new SSH user.	POST SSH Users on page 1093
/Users/	PUT	Updates an existing SSH user.	PUT SSH Users on page 1094
/Users/Access	POST	Creates a mapping from the SSH user to one or more Linux logons.	POST SSH Users Access on page 1096

2.2.25.1 SSH Keys

The SSH Keys component of the Keyfactor Web APIs 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 422: 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 below
/Unmanaged/{id}	GET	Retrieve details for a discovered unmanaged SSH key for the specified ID.	GET SSH Keys Unmanaged ID on the next page
/MyKey	GET	Retrieve details for a user's SSH key generated through Keyfactor Command.	GET SSH Keys My Key on page 958
/MyKey	POST	Generate a new SSH key pair for a user through Keyfactor Command.	POST SSH Keys My Key on page 961
/MyKey	PUT	Update an SSH key for a user through Keyfactor Command.	PUT SSH Keys My Key on page 965
Unmanaged	DELETE	Delete one or more discovered unmanaged SSH keys based on a selection query.	DELETE SSH Keys Unmanaged on page 966
Unmanaged	GET	Retrieve details for one or more discovered unmanaged SSH keys based on a selection query.	GET SSH Keys Unmanaged on page 967

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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* 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 423: DELETE SSH Keys Unmanaged {id} Input Parameters

Name	In	Description
id	Path	**Required**. 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 967) to retrieve a list of all the unmanaged keys to determine the unmanaged key's ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 424: GET SSH Keys Unmanaged {id} Input Parameters

Name	In	Description
id	Path	**Required**. 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 967) to retrieve a list of all the unmanaged keys to determine the unmanaged key's ID.

Table 425: 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	The date, in UTC, on which the SSH key was discovered.
Comments	An array 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 961). This method returns HTTP 200 OK on a success with the key's details.

Tip: The following permissions (see Security Overview) are required to use this feature:
SSH: *User* OR
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

Table 426: 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 427: GET SSH Keys My Key Response Data

Name	Description
ID	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	The date, in UTC, on which the SSH key pair was created.
StaleDate	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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 977, POST SSH Server Groups Access on page 1035, and POST SSH Servers Access on page 1002). This method returns HTTP 200 OK on a success with the key's details.

Tip: The following permissions (see Security Overview) are required to use this feature:
SSH: *User* OR
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

Table 428: 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 GET SSH Keys My Key on page 958) and encrypt it with a different password, if desired.</td></tr>
<tr><td>Comment</td><td>Body</td><td>An array 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</td></tr>
</table>

Name	In	Description
		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 429: POST SSH Keys My Key Response Data

Name	Description
ID	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	The date, in UTC, on which the SSH key pair was created.
StaleDate	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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *User* OR
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

Warning: 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 430: PUT SSH Keys My Key Input Parameters

Name	In	Description
ID	Body	**Required**. 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 life-time.
Comment	Body	An array 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 431: PUT SSH Keys My Key Response Data

Name	Description
ID	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	The date, in UTC, on which the SSH key pair was created.
StaleDate	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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* 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 432: DELETE SSH Keys Unmanaged Input Parameters

Name	In	Description
ids	Body	**Required**. An array of 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 below) to retrieve a list of all the unmanaged keys to determine the unmanaged key IDs.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 433: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Unmanaged Keys Search*. 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 434: 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	The date, in UTC, on which the SSH key was discovered.
Comments	An array 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.25.2 SSH Logons

The SSH Logons component of the Keyfactor Web APIs 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 435: SSH Logon Endpoints

Endpoint	Method	Description	Link
/{id}	DELETE	Deletes a Linux logon from Keyfactor Command.	DELETE SSH

Endpoint	Method	Description	Link
			Logons ID on the next page
/{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 973
/	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 974
/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 977

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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* 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 436: DELETE SSH Logons {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID for the SSH logon to be deleted. Use the *GET /SSH/Logons* method (see GET SSH Logons on page 973) to retrieve a list of all the SSH logons to determine the logon's ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 437: GET SSH Logons {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID for the SSH logon to retrieve. Use the *GET /SSH/Logons* method (see GET SSH Logons on page 973) to retrieve a list of all the SSH logons to determine the logon's ID.

Table 438: GET SSH Keys Unmanaged {id} 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.
Server	Details about the server on which the SSH logon resides. Server information includes: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of the server on which the SSH logon resides. Hostname — 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. UnderManagement — 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). GroupName — 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.
KeyCount	An integer indicating the number of SSH keys associated with the Linux logons.
Access	An array of key/value pairs providing information about the users mapped to the logon. Access 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 439: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Logons Search*. 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 440: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 441: 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 986) 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 1084) to retrieve a list of all the users (including service accounts) created in Keyfactor Command to determine a user's ID.

Table 442: POST 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.
Server	Details about the server on which the SSH logon resides. Server information includes: **Name** / **Description** Id — An integer indicating the Keyfactor Command reference ID of the server on which the SSH logon resides. Hostname — 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. UnderManagement — 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). GroupName — 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.
KeyCount	An integer indicating the number of SSH keys associated with the Linux logons.
Access	An array of key/value pairs providing information about the users mapped to the logon. Access 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 443: 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 973) 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 1084) 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 444: 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 key/value pairs 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.25.3 SSH Servers

The SSH Servers component of the Keyfactor Web APIs includes methods necessary to create, update, and delete SSH servers within Keyfactor Command.

Table 445: 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 the next page
/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 984
/	GET	Returns a list of a SSH servers configured in Keyfactor Command.	GET SSH Servers on page 986

Endpoint	Method	Description	Link
/	POST	Creates a new SSH server.	POST SSH Servers on page 990
/	PUT	Updates an existing SSH server.	PUT SSH Servers on page 995
/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server.	DELETE SSH Servers Access on page 1000
/Access	POST	Creates Linux logon to user and service account mappings for an SSH server.	POST SSH Servers Access on page 1002

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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 446: DELETE SSH Servers {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID for the SSH server to be deleted. Use the *GET /SSH/Servers* method (see GET SSH Servers on page 986) to retrieve a list of all the SSH servers to determine the server's ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 447: GET SSH Servers {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference ID for the SSH server to be retrieved. Use the *GET /SSH/Servers* method (see GET SSH Servers on page 986) to retrieve a list of all the SSH servers to determine the server's ID.

Table 448: 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 array 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
}
```

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

<table>
<tr><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>should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
Note: Although the 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>
</table>
</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).
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 array 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:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 449: GET SSH Servers Access {id} Input Parameters

Name	In	Description
id	Path	**Required**. 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 the next page) to retrieve a list of all the SSH servers to determine the server's ID.

Table 450: 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 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 user 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 user-name@hostname format) that has been associated with the logon.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 451: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the SSH Server Search*. 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 452: 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 array 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
}
```

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

<table>
<tr><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>should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
Note: Although the 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>
</table>
</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).
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 array 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:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 1020) and that your Keyfactor Bash Orchestrator has been registered and approved in Keyfactor Command (see GET Agents on page 12).

Tip: The following permissions (see Security Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 453: 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 454: 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 array 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
}
```

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

<table>
<tr><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>should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
Note: Although the 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>
</table>
</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).
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 array 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:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Warning: 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 455: PUT SSH Servers Input Parameters

Name	In	Description
ID	Body	**Required**. 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	The port that is configured for SSH on the SSH server. The default is 22.

Table 456: 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 array 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
}
```

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

<table>
<tr><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>should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
Note: Although the 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>
</table>
</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).
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 array 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:</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *ServerAdmin* 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 457: DELETE SSH Servers Access Input Parameters

Name	In	Description
ServerId	Body	**Required**. The Keyfactor Command reference ID for the SSH server.
LogonUsers	Body	**Required**. An array containing information for the Linux logon(s) to update. The following information should be included: **Name** / **Description** LogonName — A string indicating the name of the Linux logon. Users — An array of strings indicating the user names of one or more *users* (in DOMAIN\\username format) or *service accounts* (in username@hostname format) to be removed from association with the logon. For example: `"LogonUsers": [ { "LogonName": "johns", "Users": [ "KEYEXAMPLE\\jsmith" ] } ]`

Table 458: DELETE SSH Servers Access Response Data

Name	Description
ServerId	An integer indicating the Keyfactor Command reference ID for the SSH server.
LogonUsers	An array 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 user 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group to which the server belongs and limited for users with only the *ServerAdmin* 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 459: 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. The Keyfactor Command reference ID for the SSH server.</td></tr>
<tr><td>LogonUsers</td><td>Body</td><td>Required. An array 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 460: POST SSH Servers Access Response Data

Name	Description
ServerId	An integer indicating the Keyfactor Command reference ID for the SSH server.
LogonUsers	An array 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 user 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 user-name@hostname format) that has been associated with the logon.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.25.4 SSH Server Groups

The SSH Server Groups component of the Keyfactor Web APIs includes methods necessary to create, update and delete SSH server groups within Keyfactor Command.

Table 461: 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

Endpoint	Method	Description	Link
			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 1010
/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 1014
/	GET	Returns a list of a SSH server groups configured in Keyfactor Command.	GET SSH Server Groups on page 1015
/	POST	Creates a new SSH server group.	POST SSH Server Groups on page 1020
/	PUT	Updates an existing SSH server group.	PUT SSH Server Groups on page 1027
/Access	DELETE	Deletes Linux logon to user and service account mappings for an SSH server group.	DELETE SSH Server Groups Access on page 1034
/Access	POST	Creates Linux logon to user and service account mappings for an SSH server group.	POST SSH Server Groups Access on page 1035

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 Overview) are required to use this feature:
SSH: *EnterpriseAdmin*

Table 462: 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 1015) to retrieve a list of all the SSH server groups to determine the server group's GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 463: 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 1015) to retrieve a list of all the SSH server groups to determine the server group's GUID.

Table 464: GET SSH Server Groups {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</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>Owner</td><td>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:
<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.</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.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>A string indicating the name of the SSH server group.</td></tr>
<tr><td>SyncSchedule</td><td>An array 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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}
</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the 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 group is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
<tr><td>ServerCount</td><td>An integer indicating the number of SSH servers that belong to the server group.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 465: 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 466: GET SSH Server Groups {name} Response Data

<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>Owner</td><td>Body</td><td>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:
<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.</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.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>Body</td><td>A string indicating the name of the SSH server group.</td></tr>
<tr><td>SyncSchedule</td><td>Body</td><td>An array 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}
</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the 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>Body</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>
<tr><td>ServerCount</td><td>Body</td><td>An integer indicating the number of SSH servers that belong to the server group.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 467: GET SSH Server Groups Access {id} Input Parameters

Name	In	Description
id	Path	**Required**. 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 the next page) to retrieve a list of all the SSH server groups to determine the server group's ID.

Table 468: GET SSH Server Groups Access {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ServerGroupId</td><td>The Keyfactor Command reference GUID for the SSH server group.</td></tr>
<tr><td>LogonUsers</td><td>An array containing information for the Linux logons from the Linux server that have been stored in Keyfactor Command. Possible information includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>LogonName</td><td>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.</td></tr>
<tr><td>Users</td><td>An array of user objects containing information about the users and/or service accounts defined in Keyfactor Command that are mapped to the Linux logon. User 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 is 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 is associated with the logon.</td></tr>
</table>
Tip: Users only appear in the results if they have been mapped to the same logon on all servers in the server group.</td></tr>
</table></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 469: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Server Group Search*. 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 470: GET SSH Server Groups Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</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>Owner</td><td>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:
<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.</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.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>A string indicating the name of the SSH server group.</td></tr>
<tr><td>SyncSchedule</td><td>An array 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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the 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 group is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
<tr><td>ServerCount</td><td>An integer indicating the number of SSH servers that belong to the server group.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *EnterpriseAdmin*

Table 471: POST SSH Server Groups Input Parameters

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

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Time</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>Name</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:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>
 Note: Although the 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>
The default is unset.
</td></tr>
<tr><td>Under-Management</td><td>Body</td><td>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.</td></tr>
</table>

Table 472: POST SSH Server Groups Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ID</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>Owner</td><td>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:
<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.</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.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>A string indicating the name of the SSH server group.</td></tr>
<tr><td>SyncSchedule</td><td>An array 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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>
"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}
</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the 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 group is in inventory only mode (False) or inventory and publish policy mode (True).</td></tr>
<tr><td>ServerCount</td><td>An integer indicating the number of SSH servers that belong to the server group.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Warning: Any previously populated fields that are not submitted with their full existing data using this method will be cleared of their existing data. When using this method, you should first do a GET to retrieve all the values for the record you want to update, enter corrected data into the field(s) you want to update, and then submit all the fields using PUT, including the fields that contain values but which you are not changing.

Table 473: PUT SSH Server Groups Input Parameters

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.
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 array 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: **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. **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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>
 Note: Although the 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>
The default is unset.
</td></tr>
<tr><td>Under-Management</td><td>Body</td><td>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.</td></tr>
</table>

Table 474: PUT SSH Server Groups Response Data

<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>Owner</td><td>Body</td><td>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:
<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.</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.</td></tr>
</table></td></tr>
<tr><td>GroupName</td><td>Body</td><td>A string indicating the name of the SSH server group.</td></tr>
<tr><td>SyncSchedule</td><td>Body</td><td>An array 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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:</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</pre>
</td></tr>
</table>
Note: Although the 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>Body</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>
<tr><td>ServerCount</td><td>Body</td><td>An integer indicating the number of SSH servers that belong to the server group.</td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group and limited for users with only the *ServerAdmin* 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 475: 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. The Keyfactor Command reference ID for the SSH server group.</td></tr>
<tr><td>LogonUsers</td><td>Body</td><td>An array 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 476: DELETE SSH Server Groups Access {id} Response Data

Name	Description
ServerGroupId	The Keyfactor Command reference GUID for the SSH server group.
LogonUsers	An array 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 user 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 username@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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group and limited for users with only the *ServerAdmin* 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 477: 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. The Keyfactor Command reference ID for the SSH server group.</td></tr>
<tr><td>LogonUsers</td><td>Body</td><td>Required. An array 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 478: POST SSH Server Groups Access {id} Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>ServerGroupId</td><td>The Keyfactor Command reference GUID for the SSH server group.</td></tr>
<tr><td>LogonUsers</td><td>An array containing information for the Linux logons from the Linux server that have been stored in Keyfactor Command. Possible information includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>LogonName</td><td>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.</td></tr>
<tr><td>Users</td><td>An array of user objects containing information about the users and/or service accounts defined in Keyfactor Command that are mapped to the Linux logon. User 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 is 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 is associated with the logon.</td></tr>
</table>
Tip: Users only appear in the results if they have been mapped to the same logon on all servers in the server group.</td></tr>
</table></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.25.5 SSH Service Accounts

The SSH Service Accounts component of the Keyfactor Web APIs includes methods necessary to retrieve, create, update, rotate and delete service accounts and associated keys in Keyfactor Command.

Table 479: 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 1040
/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 1046
/	DELETE	Deletes one or more SSH service accounts with the specified IDs.	DELETE SSH Service Accounts on page 1050
/	GET	Returns a list of SSH service accounts based on the specified filters.	GET SSH Service Accounts on page 1052
/	POST	Creates a new SSH service account.	POST SSH Service Accounts on page 1059
/	PUT	Updates an existing SSH service account.	PUT SSH Service Accounts on page 1068
/Rotate/{id}	POST	Generates a new key pair for an existing service account.	POST SSH Service Accounts Rotate ID on page 1075

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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 480: 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 1052) 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: <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>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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 1046).

Tip: The following permissions (see Security Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 481: 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 1052) to retrieve a list of all the SSH service accounts to determine the service account's ID.

Table 482: GET SSH Service Accounts {id} Response Data

Name	Description
ID	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 array 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 array 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.

Name	Description
	Name / **Description**

Name	Description
Interval	A dictionary that indicates a job scheduled to run every x minutes with the specified parameter. Any interval that is selected in the UI will be converted to minutes when stored in the database. **Name** / **Description** Minutes — An integer indicating the number of minutes between each interval. For example, every hour: `"Interval": { "Minutes": 60 }`
Daily	A dictionary that indicates a job scheduled to run every day at the same time with the parameter: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }`
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** **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. 2021-05-19T16:23:01Z). 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": "2022-02-27T17:30:00Z" }` Monthly — A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z). Day — The number of the day, in the month, to run the job. For example, on the first of every month at 5:30 pm:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Monthly": {
   "Day": 1
   "Time": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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 array containing information about the service account user. Service account user 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 user.</td></tr>
<tr><td>Key</td><td>An array 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</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>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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</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 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>

Name	Description
LogonIds	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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 483: GET SSH Service Accounts Key {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 1052) 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: <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>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.

Table 484: GET SSH Service Accounts Key {id} Response Data

Name	Description
ID	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	The date, in UTC, on which the SSH key pair was created.
StaleDate	The date, in UTC, after which the SSH key pair is considered to be out of date based on the key life-time 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 485: DELETE SSH Service Accounts Input Parameters

Name	In	Description
ids	Body	**Required**. An array of 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: `{ "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" } }` 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 1046).

Tip: The following permissions (see Security Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 486: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Service Account Key Search*. The query fields supported for this endpoint are: • *CreationDate* • *Id* • *Comments* (Key comments) • *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 487: GET SSH Service Accounts Response Data

Name	Description
ID	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 array 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 array 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.

<table>
<tr><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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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** **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. 2021-05-19T16:23:01Z). 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": "2022-02-27T17:30:00Z" }` Monthly — A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z). Day — The number of the day, in the month, to run the job. For example, on the first of every month at 5:30 pm:

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

```
"Monthly": {
   "Day": 1
   "Time": "2022-02-27T17:30:00Z"
}
```

</td></tr>
</table>
Note: Although the 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 array containing information about the service account user. Service account user 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 user.</td></tr>
<tr><td>Key</td><td>An array 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</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>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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 1020) and that your Keyfactor Bash Orchestrator has been registered and approved in Keyfactor Command (see GET Agents on page 12).

Tip: The following permissions (see Security Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 488: 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 array 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 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>
</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>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 array 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. 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. username@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>
</table>

Name	In	Description
ServerGroupId	Body	**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.

Table 489: POST SSH Service Accounts Response Data

Name	Description
ID	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 array 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 array 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.

<table>
<tr><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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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**

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. 2021-05-19T16:23:01Z).
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": "2022-02-27T17:30:00Z"
}
```

Monthly

A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).
Day	The number of the day, in the month, to run the job.

For example, on the first of every month at 5:30 pm:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Monthly": {
   "Day": 1
   "Time": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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 array containing information about the service account user. Service account user 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 user.</td></tr>
<tr><td>Key</td><td>An array 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</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>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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</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 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>

Name	Description
LogonIds	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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Warning: 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 490: 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 array 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. 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>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></td></tr>
<tr><td>Id</td><td>Body</td><td>Required. The Keyfactor Command reference ID for the service account. Use the GET /SSH/ServiceAccounts method (see GET SSH Service Accounts on page 1052) to retrieve a list of all the SSH service accounts to determine the service account's ID.</td></tr>
</table>

Table 491: PUT SSH Service Accounts Response Data

Name	Description
ID	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 array 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 array 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.

<table>
<tr><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>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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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** **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. 2021-05-19T16:23:01Z). 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": "2022-02-27T17:30:00Z" }` Monthly — A dictionary that indicates a job scheduled to run on a specific day or days every month at the same time with the parameters: **Name** / **Description** Time — The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z). Day — The number of the day, in the month, to run the job. For example, on the first of every month at 5:30 pm:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>"Monthly": {
   "Day": 1
   "Time": "2022-02-27T17:30:00Z"
}</pre></td></tr>
</table>
Note: Although the 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 array containing information about the service account user. Service account user 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 user.</td></tr>
<tr><td>Key</td><td>An array 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</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>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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</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 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>

Name	Description
LogonIds	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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

SSH actions are affected by ownership on the server group with which the key is associated and limited for users with only the *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 492: 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 1052) 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: <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>It contains three IDs: • ID 2: The service account's ID. Use this one to rotate the key. • ID 7: The service account user's ID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td> 
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 493: 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.25.6 SSH Users

The SSH Users component of the Keyfactor Web APIs includes methods necessary to retrieve, create, update, rotate, and delete users and associated keys in Keyfactor Command.

Table 494: 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 below
/	GET	Returns a list of SSH users based on the specified filters.	GET SSH Users on page 1084
/	POST	Creates a new SSH user.	POST SSH Users on page 1093
/	PUT	Updates an existing SSH user.	PUT SSH Users on page 1094
/Access	POST	Creates a mapping from the SSH user to one or more Linux logons.	POST SSH Users Access on page 1096

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 Overview) are required to use this feature:
SSH: *ServerAdmin*

Table 495: DELETE SSH Users {id} Input Parameters

Name	In	Description
id	Path	**Required**. 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 1084) to retrieve a list of all the SSH users to determine the user's ID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 958) for a user account or the GET /SSH/ServiceAccounts/Key/{id} method (see GET SSH Service Accounts Key ID on page 1046) for a service account.

Tip: The following permissions (see Security Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* 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 6.

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 496: GET SSH Users {id} v2 Input Parameters

Name	In	Description
id	Path	**Required**. 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 1084) to retrieve a list of all the SSH users to determine the user's ID.

Table 497: GET SSH Users {id} v2 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 SSH user.</td></tr>
<tr><td>Key</td><td>An array containing information about the key for the 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 user'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 user.</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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</td></tr>
<tr><td>Email</td><td>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.</td></tr>
<tr><td>Comments</td><td>An array 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</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>/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 array 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 array 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 498: GET SSH Users {id} v1 Input Parameters

Name	In	Description
id	Path	**Required**. 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 1084) to retrieve a list of all the SSH users to determine the user's ID.

Table 499: GET SSH Users {id} v1 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 SSH user.</td></tr>
<tr><td>Key</td><td>An array containing information about the key for the 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 user'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 user.</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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</td></tr>
<tr><td>Email</td><td>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.</td></tr>
<tr><td>Comments</td><td>An array 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</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>/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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 958) for user accounts and the GET /SSH/ServiceAccounts/Key/{id} method (see GET SSH Service Accounts Key ID on page 1046) for service accounts.

Tip: The following permissions (see Security Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* 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 6.

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 500: 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 986) or the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 1015) 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. 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.

Name	In	Description
		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 the *Keyfactor Command Reference Guide*: *Using the SSH Server Search*. The query fields supported for this endpoint are: • *Email* • *Fingerprint* • *IsServiceAccount* • *KeyLength* • *KeyType* • *LogonCount* • *LogonServerGroupId* • *LogonServerId* • *ServiceAccountId* • *StaleDate* • *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 501: GET SSH Users v2 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 SSH user.</td></tr>
<tr><td>Key</td><td>An array containing information about the key for the 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 user'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 user.</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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</td></tr>
<tr><td>Email</td><td>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.</td></tr>
<tr><td>Comments</td><td>An array 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</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>/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 array 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 array 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 502: 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 986) or the *GET /SSH/ServerGroups* method (see GET SSH Server Groups on page 1015) 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. 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.

Name	In	Description
		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 the *Keyfactor Command Reference Guide*: *Using the SSH Server Search*.
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 503: GET SSH Users v1 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 SSH user.</td></tr>
<tr><td>Key</td><td>An array containing information about the key for the 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 user'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 user.</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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</td></tr>
<tr><td>Email</td><td>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.</td></tr>
<tr><td>Comments</td><td>An array 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</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>/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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 504: 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 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 973) to retrieve a list of all the SSH logons to determine the logon's ID(s).

Table 505: 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* role. For more information, see *SSH Permissions* in the *Keyfactor Command Reference Guide*.

Table 506: 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 1084) to retrieve a list of all the SSH users to determine the user's ID.
LogonIds	Body	An array of 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 973) 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 1084) before updating and provide both existing and new logon IDs.

Table 507: 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command 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 Overview) are required to use this feature:
SSH: *ServerAdmin* OR
SSH: *EnterpriseAdmin*

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 *ServerAdmin* 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 508: 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 1084) to retrieve a list of all the SSH users to determine the user's ID.
LogonIds	Body	An array of 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 973) 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 1084) before updating and provide both existing and new logon IDs.

Table 509: POST SSH Users Access 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 SSH user.</td></tr>
<tr><td>Key</td><td>An array containing information about the key for the 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 user'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 user.</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>The date, in UTC, on which the SSH key pair was created.</td></tr>
<tr><td>StaleDate</td><td>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.</td></tr>
<tr><td>Email</td><td>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.</td></tr>
<tr><td>Comments</td><td>An array 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</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>/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 array 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 array 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.26 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 510: SMTP Endpoints

Endpoint	Method	Description	Link
/	GET	Returns information about the SMTP configuration profile.	GET SMTP on the next page

Endpoint	Method	Description	Link
/	PUT	Updates settings for the SMTP configuration profile.	PUT SMTP on page 1101
/Test	POST	Sends a test email message to confirm SMTP configuration.	POST SMTP Test on page 1103

2.2.26.1 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 Overview) are required to use this feature:
SystemSettings: *Read*

Table 511: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.26.2 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 Overview) are required to use this feature:
SystemSettings: *Modify*

Warning: 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 512: 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 RelayAuthenticationType.</td></tr>
<tr><td>UseSSL</td><td>Body</td><td>A Boolean indicating that mail should be delivered over TLS/SSL. Not all mail servers support this.</td></tr>
</table>

Table 513: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.26.3 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 Overview) are required to use this feature:
SystemSettings: *Modify*

Table 514: 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 release.</td></tr>
<tr><td>SenderName</td><td>Body</td><td>A string indicating the name that appears as the "from" in the user's mail client (e.g. "Keyfactor Command"). This value is used for both configurations of RelayAuthenticationType.</td></tr>
</table>

Name	In	Description
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 515: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27 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 516: 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 the next page
/Endpoints/{id}	GET	Returns the details about a single endpoint discovered during SSL scanning.	GET SSL Endpoints ID on page 1112
/NetworkRanges/{id}	DELETE	Removes all network ranges from the specified SSL network.	DELETE SSL NetworkRanges ID on page 1113
/NetworkRanges/{id}	GET	Returns network range information about the specified SSL network.	GET SSL NetworkRanges ID on page 1114
/Networks/{identifier}	GET	Returns information about the specified SSL network.	GET SSL Networks Identifier on page 1115
/	GET	Returns the results of an SSL scan based on query information.	GET SSL on page 1123
/Networks	GET	Returns information about all SSL networks in Keyfactor Command.	GET SSL Networks on page 1125
/Networks	POST	Creates a new SSL network.	POST SSL Networks on page 1134
/Networks	PUT	Updates an existing SSL network.	PUT SSL Networks on page 1146
/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 1158
/Networks/{id}/Parts	GET	Returns the scan job information for SSL discovery or monitoring.	GET SSL Networks ID Parts on page 1164
/NetworkRanges	POST	Adds network ranges to the specified SSL network.	POST SSL NetworkRanges on page 1165

Endpoint	Method	Description	Link
/NetworkRanges	PUT	Updates network range information on the specified SSL network.	PUT SSL NetworkRanges on page 1166
/Endpoints/ReviewStatus	PUT	Used to change the *reviewed* status for a given SSL endpoint.	PUT SSL Endpoints Review Status on page 1167
/Endpoints/MonitorStatus	PUT	Used to change the *monitoring* status for a given SSL endpoint.	PUT SSL Endpoints Monitor Status on page 1168
/Endpoints/ReviewAll	PUT	Used to change the *reviewed* status for all given SSL endpoints to true.	PUT SSL Endpoints Review All on page 1168
/Endpoints/MonitorAll	PUT	Used to change the *monitoring* status for all given SSL endpoints to true.	PUT SSL Endpoints Monitor All on page 1169
/Networks/{id}/Scan	POST	Starts an SSL discovery or monitoring scan job manually.	POST SSL Networks ID Scan on page 1169
/NetworkRanges/Validate	POST	Validates all SSL networks given.	POST SSL NetworkRanges Validate on page 1170
/Networks/{id}	DELETE	Removes an SSL network from Keyfactor Command.	DELETE SSL Networks ID on page 1171

2.2.27.1 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 1164). This method returns HTTP 200 OK on a success with details about the specified scan job segment.

Tip: The following permissions (see Security Overview) are required to use this feature:
SslManagement: *Read*

Table 517: 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 1164) to retrieve a list of all the scan job segments in an SSL network to determine the SSL scan job segment's GUID.

Table 518: GET SSL Parts {id} Response Data

Parameter Name	Description
ScanJobPartId	The Keyfactor Command reference GUID for the scan job segment.
LogicalScanJobId	The Keyfactor Command reference GUID for the scan job as a whole.
AgentJobId	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.
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.

Parameter Name	Description
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	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	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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.2 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 519: GET SSL Endpoints {id} Input Parameters

Name	In	Description
id	Path	**Required**. The Keyfactor Command reference GUID for the SSL endpoint to be retrieved. Use the *GET /SSL* method (see GET SSL on page 1123) to retrieve a list of all the SSL endpoints to determine the SSL endpoint's GUID.

Table 520: GET SSL Endpoints {id} Response Data

Name	Description
EndpointId	The Keyfactor Command reference GUID for the endpoint.
NetworkId	The Keyfactor Command reference GUID for the SSL network that scanned the endpoint.
LastHistoryId	The Keyfactor Command reference GUID for the last history entry on the endpoint.
IpAddressBytes	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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.3 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 Overview) are required to use this feature:
SslManagement: *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 1146).

Table 521: 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 1125) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.4 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 522: 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 1125) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Table 523: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.5 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 Overview) are required to use this feature: SslManagement: *Read*

Table 524: 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 1125) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Table 525: GET SSL Networks {id} Response Data

Name	Description
NetworkId	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	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 array 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.

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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</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 array 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:</td></tr>
</table>

Name	Description
	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: **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. 2021-05-19T16:23:01Z). For example, daily at 11:30 pm: `"Daily": { "Time": "2022-02-25T23:30:00Z" }`
	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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>DiscovererPercentComplete</td><td>An integer indicating the percentage complete for a discovery job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
<tr><td>MonitorPercentComplete</td><td>An integer indicating the percentage complete for a monitoring job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
<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>
</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>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.

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.</td></tr>
<tr><td>AutoMonitor</td><td>A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).</td></tr>
<tr><td>GetRobots</td><td>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).</td></tr>
</table>

Name	Description
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 providing the list of scheduled quiet hour periods.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.6 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 526: 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 the *Keyfactor Command Reference Guide* Using the Discovery Results Search Feature section. 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 527: GET SSL Response Data

Name	Description
EndpointId	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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.7 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 528: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Network Scan Details Search*. 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 529: GET SSL Networks Response Data

Name	Description
NetworkId	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	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 array 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.

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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</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 array 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:</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>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>

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. 2021-05-19T16:23:01Z).</td></tr>
</table>

For example, daily at 11:30 pm:

<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Immediate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>DiscovererPercentComplete</td><td>An integer indicating the percentage complete for a discovery job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
<tr><td>MonitorPercentComplete</td><td>An integer indicating the percentage complete for a monitoring job. The percentage complete will be zero for small jobs for the entire duration of the job because this value is updated upon completion of each segment of a scan job (and small jobs generally consist of only one segment). All jobs will show 100% at completion. The counter resets when a new job begins.</td></tr>
<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>
</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>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.

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.</td></tr>
<tr><td>AutoMonitor</td><td>A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).</td></tr>
<tr><td>GetRobots</td><td>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).</td></tr>
</table>

Name	Description
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 providing the list of scheduled quiet hour periods.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.8 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 530: POST SSL Networks Input Parameters

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>NetworkId</td><td>Body</td><td>The Keyfactor Command reference GUID for the SSL network. This GUID is automatically set by Keyfactor Command.</td></tr>
<tr><td>Name</td><td>Body</td><td>Required. A string indicating the name for the SSL network.</td></tr>
<tr><td>AgentPoolName</td><td>Body</td><td>Required. A string indicating the name of the orchestrator pool assigned to the SSL network. See in the Keyfactor Command Reference Guide for more information.</td></tr>
<tr><td>AgentPoolId</td><td>Body</td><td>The Keyfactor Command reference GUID for the orchestrator pool assigned to the SSL network.</td></tr>
<tr><td>Description</td><td>Body</td><td>Required. A string indicating the description of the SSL network.</td></tr>
<tr><td>Enabled</td><td>Body</td><td>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.</td></tr>
<tr><td>DiscoverSchedule</td><td>Body</td><td>An array 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:
<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>
<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></td><td><pre>}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
<tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. "Sunday").</td></tr>
</table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",</pre></td></tr>
</table>
</td></tr>
</table>

<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>
      "Wednesday",
      "Friday"
   ],
   "Time": "2022-02-27T17:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>
"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}
</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>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>Body</td><td>An array 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>
<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. 2021-05-19T16:23:01Z).</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>For example, daily at 11:30 pm:
<pre>"Daily": {
  "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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</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>and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</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. This field is for reference and is not configurable.</td></tr>
<tr><td>Monit-</td><td>Body</td><td>An integer indicating the percentage complete for a monitoring job. The percentage</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>orPercentComplete</td><td></td><td>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-erLastScanned</td><td>Body</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. This field is for reference and is not configurable.</td></tr>
<tr><td>MonitorLastScanned</td><td>Body</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>
</table>

Name	In	Description
		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.
AutoMonitor	Body	A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).
GetRobots	Body	A Boolean that indicates whether orchestrators should perform a GET /robots.txt request during scans in order to behave like a webcrawler and provide an explanation of network activity (true) or not (false).
DiscoverTimeoutMs	Body	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however they will also increase the chance of missing a certificate on a slow or congested network.
MonitorTimeoutMs	Body	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
ExpirationAlertDays	Body	An integer that indicates the number of days within which to begin providing warnings regarding upcoming expiration in notification email messages.
DiscoverJobParts	Body	An integer that indicates the number of job parts that have been created for a discovery job. This field is for reference and is not configurable.
MonitorJobParts	Body	An integer that indicates the number of job parts that have been created for a monitoring job. This field is for reference and is not configurable.
QuietHours	Body	An array providing the list of scheduled quiet hour periods. For example: `"QuietHours": [ { "StartDay": "Monday", "StartTime": "2022-11-21T14:00:08Z", "EndDay": "Tuesday", "EndTime": "2022-11-22T14:00:08Z" }, { "StartDay": "Saturday", "StartTime": "2022-11-26T04:00:08Z", "EndDay": "Sunday",`

Name	In	Description
		<pre>"EndTime": "2022-11-27T16:00:08Z" }]</pre>

Table 531: POST SSL Networks Response Data

Name	Description
NetworkId	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	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 array providing the discovery schedule for the SSL network group.
MonitorSchedule	An array 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: **Value** / **Description** 0 – Unknown 1 – Not Scheduled 2 – Running

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

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.</td></tr>
<tr><td>AutoMonitor</td><td>A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).</td></tr>
<tr><td>GetRobots</td><td>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</td></tr>
</table>

Name	Description
	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 providing the list of scheduled quiet hour periods.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.9 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 Overview) are required to use this feature:
SslManagement: *Modify*

Warning: 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 532: PUT SSL Networks Input Parameters

Name	In	Description
NetworkId	Body	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 in the *Keyfactor Command Reference Guide* for more information.
AgentPoolId	Body	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 array 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.

For example, every hour:

```
"Interval": {
   "Minutes": 60
```

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td><pre>}</pre></td></tr>
<tr><td>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. 2021-05-19T16:23:01Z).</td></tr></table>
For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr><tr><td>Days</td><td>An array of values representing the days of the week on which to run the job. These can either be entered as integers (0 for Sunday, 1 for Monday, etc.) or as days of the week (e.g. "Sunday").</td></tr></table>
For example, every Monday, Wednesday and Friday at 5:30 pm:
<pre>"Weekly": {
   "Days": [
      "Monday",</pre></td></tr>
</table>
</td></tr>
</table>

<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>
      "Wednesday",
      "Friday"
   ],
   "Time": "2022-02-27T17:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}
</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>
"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}
</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>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>Body</td><td>An array 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>
<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. 2021-05-19T16:23:01Z).</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>For example, daily at 11:30 pm:
<pre>"Daily": {
   "Time": "2022-02-25T23:30:00Z"
}</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. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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</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>and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).</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": "2022-02-27T17:30:00Z"
}</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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</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. This field is for reference and is not configurable.</td></tr>
<tr><td>Monit-</td><td>Body</td><td>An integer indicating the percentage complete for a monitoring job. The percentage</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>orPercentComplete</td><td></td><td>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-erLastScanned</td><td>Body</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. This field is for reference and is not configurable.</td></tr>
<tr><td>MonitorLastScanned</td><td>Body</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>
</table>

Name	In	Description
		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.
AutoMonitor	Body	A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).
GetRobots	Body	A Boolean that indicates whether orchestrators should perform a GET /robots.txt request during scans in order to behave like a webcrawler and provide an explanation of network activity (true) or not (false).
DiscoverTimeoutMs	Body	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to discover the endpoint. Shorter timeout periods will increase the overall scanning throughput, however they will also increase the chance of missing a certificate on a slow or congested network.
MonitorTimeoutMs	Body	An integer that indicates the amount of time (in milliseconds) the scan will wait (before timing out) to receive the discovered endpoint certificate expiration details.
ExpirationAlertDays	Body	An integer that indicates the number of days within which to begin providing warnings regarding upcoming expiration in notification email messages.
DiscoverJobParts	Body	An integer that indicates the number of job parts that have been created for a discovery job. This field is for reference and is not configurable.
MonitorJobParts	Body	An integer that indicates the number of job parts that have been created for a monitoring job. This field is for reference and is not configurable.
QuietHours	Body	An array providing the list of scheduled quiet hour periods. For example: `"QuietHours": [ { "StartDay": "Monday", "StartTime": "2022-11-21T14:00:08Z", "EndDay": "Tuesday", "EndTime": "2022-11-22T14:00:08Z" }, { "StartDay": "Saturday", "StartTime": "2022-11-26T04:00:08Z", "EndDay": "Sunday",`

Name	In	Description
		"EndTime": "2022-11-27T16:00:08Z" }]

Table 533: PUT SSL Networks Response Data

Name	Description
NetworkId	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	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 array providing the discovery schedule for the SSL network group.
MonitorSchedule	An array 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: **Value** / **Description** 0 – Unknown 1 – Not Scheduled 2 – Running

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

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.</td></tr>
<tr><td>AutoMonitor</td><td>A Boolean that indicates whether automatic monitoring of discovered endpoints is enabled (true) or not (false).</td></tr>
<tr><td>GetRobots</td><td>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</td></tr>
</table>

Name	Description
	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 providing the list of scheduled quiet hour periods.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.10 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 534: 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 1123) 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 535: GET SSL Endpoints {id} History Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>HistoryId</td><td>The Keyfactor Command reference GUID for the history entry.</td></tr>
<tr><td>EndpointId</td><td>The Keyfactor Command reference GUID for the endpoint with which the history is associated.</td></tr>
<tr><td>AuditId</td><td>The Keyfactor Command ID used to track progress during scan jobs.</td></tr>
<tr><td>Timestamp</td><td>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. The 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>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 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>The date, in UTC, on which the certificate was issued by the certificate authority.</td></tr>
<tr><td>NotAfter</td><td>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>Thumbprint</td><td>A string indicating the thumbprint of the certificate.</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></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 — An array 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.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.11 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 536: 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 1125) 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Network Scan Details Search*.
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 537: 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	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	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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.12 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 538: POST SSL Network Ranges Input Parameters

Name	In	Description
NetworkId	Body	**Required**. The Keyfactor Command reference GUID for the SSL network. Use the GET /SSL/Networks method (see GET SSL Networks on page 1125) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.13 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 Overview) are required to use this feature:
SslManagement: *Modify*

Warning: 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 539: PUT SSL Network Ranges {id} Input Parameters

Name	In	Description
NetworkId	Body	**Required**. The Keyfactor Command reference GUID for the SSL network. Use the GET /SSL/Networks method (see GET SSL Networks on page 1125) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.14 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 540: 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 1123) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.15 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 541: 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 1123) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.16 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 542: 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 the *Keyfactor Command Reference Guide*Using the Discovery Results Search Feature section.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.17 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 543: 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 the *Keyfactor Command Reference Guide* Using the Discovery Results Search Feature section.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.18 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 544: 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 1125) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.19 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 545: 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 1125) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.20 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 Overview) are required to use this feature:
SslManagement: *Read*

Table 546: POST SSL Network Ranges Validate Input Parameters

Name	In	Description
networkRangesToVerify	Body	**Required**. An array of network ranges to validate. For example: `["10.5.4.0/24:443","192.168.12.0/16:443,22","keyexample.com:443"]`

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.27.21 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 Overview) are required to use this feature:
SslManagement: *Modify*

Table 547: 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 1125) to retrieve a list of all the SSL networks to determine the SSL network's GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.28 Status

The Status component of the Keyfactor API includes methods necessary to retrieve the current list of Keyfactor API endpoints.

Table 548: Status Endpoints

Endpoint	Method	Description	Link
/Endpoints	GET	Returns a list of the Keyfactor API endpoints.	GET Status Endpoints below

2.2.28.1 GET Status Endpoints

The GET /Status/Endpoints method returns a list of all the endpoints currently available for use in the Keyfactor API. There are no input parameters for this method. 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 Overview) are required to use this feature:
None

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.29 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 549: Templates Endpoints

Endpoint	Method	Description	Link
/{id}	GET	Returns information about the specified template.	GET Templates ID on the next page
/Settings	GET	Returns the global template policy settings.	GET Templates Settings on page 1186
/Settings	PUT	Sets global values for template policy.	PUT Templates Settings on page 1192
/SubjectParts	GET	Returns a list of supported subject parts for template regular expressions and default subjects.	GET Templates Subject Parts on page 1205
/	GET	Returns a list of templates.	GET Templates on page 1206
/	PUT	Updates selected settings for the specified template.	PUT Templates on page 1216

Endpoint	Method	Description	Link
/Import	POST	Import templates from a specified configuration tenant into Keyfactor Command	POST Templates/Import on page 1243

2.2.29.1 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 Overview) are required to use this feature:
PkiManagement: *Read*

Table 550: 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 1206) to retrieve a list of all the templates to determine the template ID.

Table 551: 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 defined by the CA. The field is not configurable.
KeyType	A string indicating the key type of the template as defined by the CA. 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.
FriendlyName	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.
KeyRetention	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>Name</th><th>Description</th></tr>
<tr><td></td><td>
<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 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>
</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>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 object 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:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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>
</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>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:
<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".
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 user enters information in a metadata field that does not match the template-specific regular expression (Validation field).</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</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>choice.</td></tr>
</table>
</td></tr>
<tr><td>AllowedEn-rollmentTypes</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>
<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 object 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 1186. The template regular expression object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>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>
</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>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 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the organization name entered in the field be one of "Key Example Inc", "Key Example" or "Key Example Inc.":
<code>^(?:Key Example Inc|Key Example|Key Example, Inc\.)$</code>
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.</td></tr>
<tr><td>OU (Organization Unit)</td><td>This regular expression requires that the organizational unit entered in the field be one of these four departments:</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

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$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative	This regular expression specifies that the data

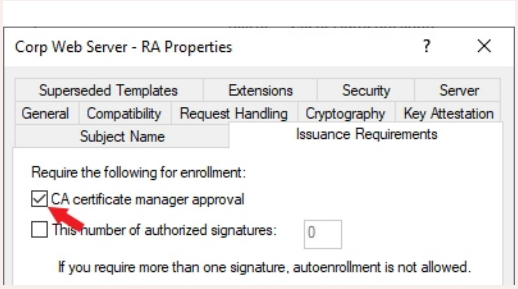
Name	Description
	Name / **Description**
	Error: A string specifying the error message displayed to the user when the subject part referenced in the CSR or entered for a PFX enrollment does not match

Subject Part / **Example**

Subject Part	Example
Name: IPv4 Address)	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 upper-case 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$`

<table>
<tr><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 given regular expression. Note that 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 object 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 GET Templates Settings on page 1186. 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 1205) 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 1186. The template policy object contains the following parameters:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TempalteId</td><td>The Keyfactor Command reference ID of the certificate template the policy is associated with.</td></tr>
<tr><td>RSAValidKeySizes</td><td>An object containing a comma-delimited list of integers defining the valid RSA key sizes supported for all templates used for enrollment. The supported values are:
2048
4096</td></tr>
<tr><td>ECCValidCurves</td><td>An object containing a list of strings defining the valid elliptic curve</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>algorithms for ECC templates. These 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>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>
<tr><td>AllowEd448</td><td>A Boolean that indicates whether Ed448 key type is allowed (true) or not (false).</td></tr>
<tr><td>AllowEd25519</td><td>A Boolean that indicates whether Ed25519 key type is allowed (true) or not (false).</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 on a Microsoft CA. In addition to granting permissions at the template level, you need enable the</td></tr>
</table>

Name	Description
	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 object 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 2: Microsoft Issuance Requirements on a Template for Manager Approval*
KeyUsage	An integer indicating the total key usage of the certificate. Key usage is stored in Active Directory as a single value made of a combination of values. The values that make up the key usage value include: **Value / Function / Description** 0 / None / No key usage parameters. 1 / Encipherment Only / The key can be used for encryption only. 2 / CRL Signing / The key can be used to sign a certificate revocation list (CRL).

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></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>ExtendedKeyUsages</td><td>An object containing the extended key usage information for the template. This field is populated from the CA and is not configurable. The extended key usage object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the extended key usage in Active Directory.</td></tr>
<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 OID of the elliptic curve algorithm configured for the template, 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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.29.2 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.

Tip: Template policies may also be set at an individual template level to apply to a single template (see PUT Templates on page 1216). Template policies set at the individual template level take precedence over template policies set at the global level.

Tip: The following permissions (see Security Overview) are required to use this feature:
PkiManagement: *Read*

There are no input parameters for this method.

Table 552: GET Templates Settings Response Data

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TemplateRegexes</td><td>An object 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:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see <a>GET Templates Subject Parts on page 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_
\.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the organization name entered in the field be one of "Key Example Inc", "Key Example" or "Key Example Inc.":
<code>^(?:Key Example Inc|Key Example|Key Example, Inc\.)$</code>
The period in the final company name (Key Example,</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Value	Description
	Name / **Description**

Subject Part	Example
	Inc.) needs to be escaped in the regular expression with a slash ("\") but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com":

Value	Description
	Name / **Description**

Subject Part	Example
	`^[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}$`
IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
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 lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
UPN (Subject Alternative Name: User Principal Name)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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 that 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 object 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>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).</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": [
   {
      "SubjectPart": "L",
      "Value": "Denver"
   },
   {
      "SubjectPart": "ST",
      "Value": "Colorado"
   }
]
</pre>
Note: See also the Subject Format application setting, which takes precedence over enroll-
</td></tr>
</table>

Value	Description
	ment 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 array 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:

Value	Description
RSAValidKeySizes	An object containing a comma-delimited list of integers defining the valid RSA key sizes supported for all templates used for enrollment. The supported values are: • 2048 • 4096
ECCValidCurves	An object containing a list of strings defining the valid elliptic curve algorithms for ECC templates. These 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").
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.

Value	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>AllowEd448</td><td>A Boolean that indicates whether Ed448 key type is allowed (true) or not (false).</td></tr><tr><td>AllowEd25519</td><td>A Boolean that indicates whether Ed25519 key type is allowed (true) or not (false).</td></tr></table> For example: <pre>"TemplatePolicy": { "RSAValidKeySizes": [2048, 4096], "ECCValidCurves": ["1.2.840.10045.3.1.7", "1.3.132.0.34" "1.3.132.0.35"], "AllowKeyReuse": false, "AllowWildcards": true, "RFCEnforcement": true, "AllowEd448": false, "AllowEd25519": false }</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.29.3 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 1216). 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 Overview) are required to use this feature:
PkiManagement: *Modify*

Warning: 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 553: PUT Templates Settings Input Parameters

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TemplateRegexes</td><td>An object 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:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see <a>GET Templates Subject Parts on page 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_
\.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the organization name entered in the field be one of "Key Example Inc", "Key Example" or "Key Example Inc.":
<code>^(?:Key Example Inc|Key Example|Key Example, Inc\.)$</code>
The period in the final company name (Key Example,</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Value	Description
	Name / **Description**

Subject Part	Example
	Inc.) needs to be escaped in the regular expression with a slash ("\") but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com":

Value	Description
	Name / **Description**

Subject Part	Example
	`^[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}$`
IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
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 lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
UPN (Subject Alternative Name: User Principal Name)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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 that 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 object 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>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).</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": [
   {
      "SubjectPart": "L",
      "Value": "Denver"
   },
   {
      "SubjectPart": "ST",
      "Value": "Colorado"
   }
]
</pre>
Note: See also the Subject Format application setting, which takes precedence over enroll-
</td></tr>
</table>

Value	Description
	ment 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 array 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:

Value	Description
RSAValidKeySizes	An object containing a comma-delimited list of integers defining the valid RSA key sizes supported for all templates used for enrollment. The supported values are: • 2048 • 4096
ECCValidCurves	An object containing a list of strings defining the valid elliptic curve algorithms for ECC templates. These 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").
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.

Value	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>AllowEd448</td><td>A Boolean that indicates whether Ed448 key type is allowed (true) or not (false).</td></tr><tr><td>AllowEd25519</td><td>A Boolean that indicates whether Ed25519 key type is allowed (true) or not (false).</td></tr></table> For example: <pre>"TemplatePolicy": { "RSAValidKeySizes": [2048, 4096], "ECCValidCurves": ["1.2.840.10045.3.1.7", "1.3.132.0.34" "1.3.132.0.35"], "AllowKeyReuse": false, "AllowWildcards": true, "RFCEnforcement": true, "AllowEd448": false, "AllowEd25519": false }</pre>

Table 554: PUT Templates Settings Response Data

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TemplateRegexes</td><td>An object 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:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>SubjectPart</td><td>A string indicating the portion of the subject the regular expression applies to (e.g. CN).</td></tr>
<tr><td>RegEx</td><td>A string specifying the regular expression against which data entered in the indicated subject part field (e.g. CN) in the enrollment pages of the Keyfactor Command Management Portal or using an API enrollment method will be validated.
Use the GET /Templates/SubjectParts method (see GET Templates Subject Parts on page 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_
\.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the organization name entered in the field be one of "Key Example Inc", "Key Example" or "Key Example Inc.":
<code>^(?:Key Example Inc|Key Example|Key Example, Inc\.)$</code>
The period in the final company name (Key Example,</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Value	Description
	Name / **Description**

Subject Part	Example
	Inc.) needs to be escaped in the regular expression with a slash ("\") but the comma does not.
OU (Organization Unit)	This regular expression requires that the organizational unit entered in the field be one of these four departments: `^(?:IT|HR|Accounting|E-Commerce)$`
L (City/Locality)	This regular expression requires that the city entered in the field be one of these five cities: `^(?:Boston|Chicago|New York|London|Dallas)$`
ST (State/Province)	This regular expression requires that the state entered in the field be one of these eight states: `^(?:Massachusetts|Illinois|New York|Ontario|Texas)$`
C (Country)	This regular expression requires that the country entered in the field be either US or CA: `^(?:US|CA)$`
E (Email)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com":

Value	Description
	Name / **Description**

Subject Part	Example
	`^[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}$`
IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or uppercase letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
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 lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`
UPN (Subject Alternative Name: User Principal Name)	This regular expression specifies that the data entered in the field must consist of some number of characters prior to the "@" made up only of lower-case letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$`

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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 that 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 object 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>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).</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": [
   {
      "SubjectPart": "L",
      "Value": "Denver"
   },
   {
      "SubjectPart": "ST",
      "Value": "Colorado"
   }
]
</pre>
Note: See also the Subject Format application setting, which takes precedence over enroll-
</td></tr>
</table>

Value	Description
	ment 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 array 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:

Value	Description
RSAValidKeySizes	An object containing a comma-delimited list of integers defining the valid RSA key sizes supported for all templates used for enrollment. The supported values are: • 2048 • 4096
ECCValidCurves	An object containing a list of strings defining the valid elliptic curve algorithms for ECC templates. These 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").
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.

Value	Description
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>AllowEd448</td><td>A Boolean that indicates whether Ed448 key type is allowed (true) or not (false).</td></tr><tr><td>AllowEd25519</td><td>A Boolean that indicates whether Ed25519 key type is allowed (true) or not (false).</td></tr></table> For example: <pre>"TemplatePolicy": { "RSAValidKeySizes": [2048, 4096], "ECCValidCurves": ["1.2.840.10045.3.1.7", "1.3.132.0.34" "1.3.132.0.35"], "AllowKeyReuse": false, "AllowWildcards": true, "RFCEnforcement": true, "AllowEd448": false, "AllowEd25519": false }</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.29.4 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.

Tip: The following permissions (see Security Overview) are required to use this feature:
PkiManagement: *Read*

Table 555: 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.29.5 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 Overview) are required to use this feature:
PkiManagement: *Read*

Table 556: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Template Search Feature*. The query fields supported for this endpoint are: • *AllowedEnrollmentType* (1-PFX Enrollment, 2-CSR Enrollment, 3-CSR Generation, `0`-None) • *DisplayName* • *FriendlyName* • *ForestRoot* (deprecated) • *ConfigurationTenant* • *HasPrivateKeyRetention* (True, False) • *IsDefaultTemplate* (True, False) • *KeyType* (Unknown, RSA, DSA, ECC, DH) • *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*.
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 557: 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 defined by the CA. The field is not configurable.
KeyType	A string indicating the key type of the template as defined by the CA. 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.
FriendlyName	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.
KeyRetention	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>Name</th><th>Description</th></tr>
<tr><td></td><td>
<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 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>
</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>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>AllowedEn-rollmentTypes</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>
<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 object 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</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>regular expressions. For more information about system-wide regular expressions, see GET Templates Settings on page 1186. The template regular expression object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>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 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_
\.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the organization name entered in the field be one of "Key Example Inc", "Key Example" or "Key Example Inc.":</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Name	Description
	Name / **Description**

Subject Part	Example
	`^(?: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: `^(?: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	This regular expression specifies that the data entered in the field must consist of some number

Name	Description
	Name / **Description**

Subject Part	Example
Name)	of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative Name: IPv4 Address)	This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$` This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods: `^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`
IPv6 (Subject Alternative Name: IPv6 Address)	This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or upper-case letters separated by colons: `^(?:[A-F0-9]{1,4}:){7}[A-F0-9]{1,4}$`
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>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 that 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>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 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.</td></tr>
<tr><td>AllowedRequesters</td><td>An object containing the list of Keyfactor Command security roles—as strings—that have been granted enroll permission on the template.</td></tr>
<tr><td>DisplayName</td><td>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.</td></tr>
<tr><td>RequiresApproval</td><td>A Boolean indicating whether the template has been configured with the Microsoft CA certificate manager approval option enabled (true) or not (false).</td></tr>
</table>

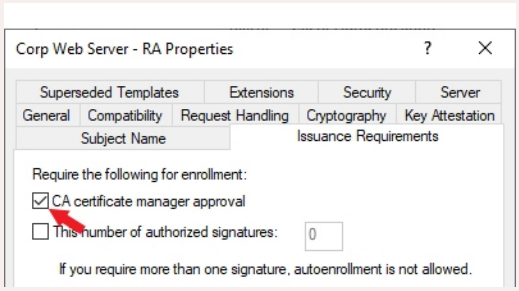
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>Important: Any templates that are configured on the Microsoft CA Issuance Requirements tab for CA certificate manager approval cannot be used for enrollment and associated alerting in Keyfactor Command without configuring private key 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 3: Microsoft Issuance Requirements on a Template for Manager Approval</td></tr>
<tr><td>KeyUsage</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 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>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></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>ExtendedKeyUsages</td><td>An object containing the extended key usage information for the template. This field is populated from the CA and is not configurable. The extended key usage object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the extended key usage in Active Directory.</td></tr>
<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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.29.6 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 Overview) are required to use this feature:
PkiManagement: *Modify*

Warning: 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 558: 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>KeySize</td><td>Body</td><td>A string indicating the minimum supported key size of the template as defined by the CA. The field is not configurable.</td></tr>
<tr><td>KeyType</td><td>Body</td><td>A string indicating the key type of the template as defined by the CA. The field is not configurable.</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:</td></tr>
</table>

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

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 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": [
   {
      "Id": 3,
      "Name": "MyCustomField",
      "Options": ["Green","Red","Yellow","Blue"],
      "DataType": 2
   }
</pre>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td><pre>]</pre></td></tr>
<tr><td>MetadataFields</td><td>Body</td><td>An object 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:<table><tr><th>Name</th><th>Description</th></tr><tr><td>Id</td><td>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".</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>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 user enters information in a metadata field that does not match the template-specific regular expression (Validation field).</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>
For example:
<pre>
"MetadataFields": [
  {
    "Id": 4,
    "DefaultValue": "reggie.wallace@keyexample.com",
</pre>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<pre>
      "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>Body</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>
<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 object containing individual template-level regular expressions against which to validate the subject data. Regular expressions defined on a template apply to enroll-</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>ments 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 1186. The template regular expression object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>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 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organ-</td><td>This regular expression requires that the</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Name	In	Description
		Name / **Description** **Subject Part** / **Example** ization) — 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: `^(?: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,

Name	In	Description
		Name / **Description** **Subject Part** / **Example** periods, and/or hyphens followed by exactly "@keyexample.com": `^[a-zA-Z0-9'_\.\-]*@keyexample\.com$` DNS (Subject Alternative Name: DNS Name) — This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com\|keyexample2\.com)$` IPv4 (Subject Alternative Name: IPv4 Address) — This regular expression specifies that the data entered in the field must be exactly "130.101." followed by anywhere between 1 and 3 numbers followed by exactly "." followed by anywhere between 1 and 3 numbers: `^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$` This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods: `^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$` 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>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>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. Note that 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 object containing individual template-level template default settings. Template</td></tr>
</table>

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>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 GET Templates Settings on page 1186. 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 1205) 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": [
   {
      "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 1186. The template policy object contains the following parameters:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TempalteId</td><td>The Keyfactor Command reference ID of the certificate template the policy is associated with.</td></tr>
<tr><td>RSAValidKeySizes</td><td>An object containing a comma-delimited list of integers defining the valid RSA key sizes supported for all templates used for enrollment. The supported values are:</td></tr>
</table>
</td></tr>
</table>

Name	In	Description
		<table><tr><th>Value</th><th>Description</th></tr><tr><td></td><td> 2048 4096 </td></tr><tr><td>ECCValidCurves</td><td>An object containing a list of strings defining the valid elliptic curve algorithms for ECC templates. These 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>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><tr><td>AllowEd448</td><td>A Boolean that indicates whether Ed448 key type is allowed (true) or not (false).</td></tr><tr><td>AllowEd25519</td><td>A Boolean that indicates whether Ed25519 key type is allowed (true) or not (false).</td></tr></table> For example:

Name	In	Description
		<pre>"TemplatePolicy": { "TemplateId": 17, "RSAValidKeySizes": [2048, 4096], "ECCValidCurves": ["1.2.840.10045.3.1.7", "1.3.132.0.34" "1.3.132.0.35"], "AllowKeyReuse": false, "AllowWildcards": true, "RFCEnforcement": true, "AllowEd448": false, "AllowEd25519": false }</pre>
UseAl-lowedRequesters	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 object 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>
RequiresApproval	Body	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

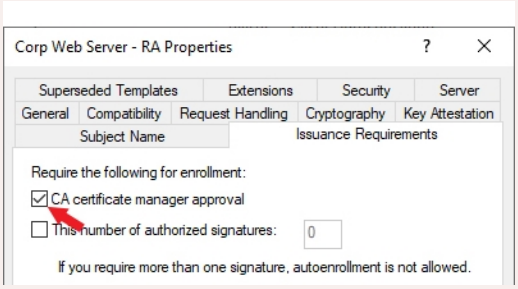
<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td></td><td></td><td>Requirements tab for CA certificate manager approval cannot be used for enrollment and associated alerting in Keyfactor Command without configuring private key retention. Any of the enabled private key retention settings (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 4: 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 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></td></tr>
</table>

Name	In	Description
		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.
Curve	Body	A string indicating the OID of the elliptic curve algorithm configured for the template, 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

Table 559: 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 defined by the CA. The field is not configurable.
KeyType	A string indicating the key type of the template as defined by the CA. 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.
FriendlyName	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.
KeyRetention	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>Name</th><th>Description</th></tr>
<tr><td></td><td>
<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 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>
</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>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 object 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:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>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>
</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>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:
<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".
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 user enters information in a metadata field that does not match the template-specific regular expression (Validation field).</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</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>choice.</td></tr>
</table>
</td></tr>
<tr><td>AllowedEn-rollmentTypes</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>
<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 object 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 1186. The template regular expression object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>TemplateId</td><td>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>
</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>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 1205) to retrieve a list of all the supported subject parts.
The following are some regular expression examples:
<table>
<tr><th>Subject Part</th><th>Example</th></tr>
<tr><td>CN (Common Name)</td><td>This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly ".keyexample.com":
<code>^[a-zA-Z0-9'_ \.\-]*\.keyexample\.com$</code>
The default value for the Common Name regular expression is:
<code>.+</code>
This requires entry of at least one character in the Common Name field in the enrollment pages.</td></tr>
<tr><td>O (Organization)</td><td>This regular expression requires that the organization name entered in the field be one of "Key Example Inc", "Key Example" or "Key Example Inc.":
<code>^(?:Key Example Inc|Key Example|Key Example, Inc\.)$</code>
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.</td></tr>
<tr><td>OU (Organization Unit)</td><td>This regular expression requires that the organizational unit entered in the field be one of these four departments:</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

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$`
DNS (Subject Alternative Name: DNS Name)	This regular expression specifies that the data entered in the field must consist of some number of characters in the first portion of the field made up only of lowercase letters, uppercase letters, numbers, apostrophes, underscores, periods, and/or hyphens followed by exactly either ".keyexample1.com" or ".keyexample2.com": `^[a-zA-Z0-9'_\.\-]*\.(?:keyexample1\.com|keyexample2\.com)$`
IPv4 (Subject Alternative	This regular expression specifies that the data

<table>
<tr><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>Name: IPv4 Address)</td><td>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:
<code>^130\.101\.(?:[0-9]{1,3})\.(?:[0-9]{1,3})$</code>
This regular expression specifies only that the IPv4 address is made up of 4 sets of between 1 and 3 numbers separated by periods:
<code>^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$</code></td></tr>
<tr><td>IPv6 (Subject Alternative Name: IPv6 Address)</td><td>This regular expression specifies that the data entered in the field must be made up of eight sets of between one and four numbers and/or upper-case 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</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>the given regular expression. Note that 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 object 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 GET Templates Settings on page 1186. 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 1205) 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 1186. The template policy object contains the following parameters:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>TempalteId</td><td>The Keyfactor Command reference ID of the certificate template the policy is associated with.</td></tr>
<tr><td>RSAValidKeySizes</td><td>An object containing a comma-delimited list of integers defining the valid RSA key sizes supported for all templates used for enrollment. The supported values are:
2048
4096</td></tr>
<tr><td>ECCValidCurves</td><td>An object containing a list of strings defining the valid elliptic curve</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>algorithms for ECC templates. These 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>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>
<tr><td>AllowEd448</td><td>A Boolean that indicates whether Ed448 key type is allowed (true) or not (false).</td></tr>
<tr><td>AllowEd25519</td><td>A Boolean that indicates whether Ed25519 key type is allowed (true) or not (false).</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 on a Microsoft CA. In addition to granting permissions at the template level, you need enable the</td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>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.</td></tr>
<tr><td>AllowedRequesters</td><td>An object containing the list of Keyfactor Command security roles—as strings—that have been granted enroll permission on the template.</td></tr>
<tr><td>DisplayName</td><td>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.</td></tr>
<tr><td>RequiresApproval</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.

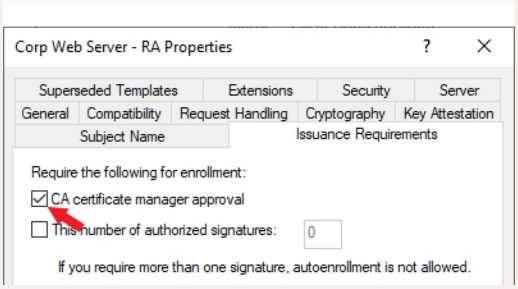
 

Figure 5: Microsoft Issuance Requirements on a Template for Manager Approval</td></tr>
<tr><td>KeyUsage</td><td>An integer indicating the total key usage of the certificate. Key usage is stored in Active Directory as a single value made of a combination of values. The values that make up the key usage 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>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Function</th><th>Description</th></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>ExtendedKeyUsages</td><td>An object containing the extended key usage information for the template. This field is populated from the CA and is not configurable. The extended key usage object contains the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Id</td><td>An integer indicating the ID of the extended key usage in Active Directory.</td></tr>
<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 OID of the elliptic curve algorithm configured for the template, 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>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.29.7 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 Overview) are required to use this feature:
PkiManagement: *Modify*

Table 560: POST Templates/Import Input Parameters

Name	Description
ConfigurationTenant	A string indicating the name of the configuration tenant from which to import.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.30 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 1259 and Workflow Instances on page 1354).

Table 561: Workflow Certificates Endpoints

Endpoint	Method	Description	Link
/Certificates/{id}	GET	Retrieve certificate request information for a single request.	GET Workflow Certificates ID on the next page
/Certificates/Denied	GET	Retrieve a list of denied certificate request(s).	GET Workflow Certificates Denied on page 1246

Endpoint	Method	Description	Link
/Certificates/Pending	GET	Retrieve a list of outstanding pending certificate request(s).	GET Workflow Certificates Pending on page 1249
/Certificates/ExternalValidation	GET	Retrieve a list of certificate request(s) requiring external validation.	GET Workflow Certificates External Validation on page 1252
/Certificates/Approve	POST	Approve a list of pending certificate request(s).	POST Workflow Certificates Approve on page 1257
/Certificates/Deny	POST	Deny a list of pending certificate request(s).	POST Workflow Certificates Deny on page 1255

2.2.30.1 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 1259 and Workflow Instances on page 1354).

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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 562: 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 1249) to retrieve a list of all the certificate requests to determine the certificate request ID.

Table 563: GET Workflow Certificates {id} Input Parameters

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>DenialComment</td><td>A string containing the user-provided comment entered when the certificate request was denied.</td></tr>
<tr><td>KeyLength</td><td>An integer indicating the key length of the certificate request.</td></tr>
<tr><td>SANs</td><td>An object containing a comma delimited list of strings listing the subject alternative name elements of the certificate request.</td></tr>
<tr><td>CertStores</td><td>An object containing the certificate store locations to which the certificate resulting from the request will be distributed once approved. Certificate store location data includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>EntryName</td><td>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.</td></tr>
<tr><td>ClientMachine</td><td>A string indicating the machine on which the certificate store is located.</td></tr>
<tr><td>StorePath</td><td>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.</td></tr>
</table></td></tr>
<tr><td>Curve</td><td>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</td></tr>
<tr><td>Id</td><td>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.</td></tr>
<tr><td>CARequestId</td><td>An integer indicating the row index of the certificate request in the certificate authority.</td></tr>
<tr><td>CommonName</td><td>A string indicating the common name of the requested certificate.</td></tr>
</table>

Name	Description
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).
Metadata	An array containing the metadata fields populated for the certificate request.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.30.2 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 1259 and Workflow Instances on page 1354).

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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 564: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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 "2022-09-01T00:00:00Z" 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 565: 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 array containing the metadata fields populated for the certificate request.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.30.3 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 1259 and Workflow Instances on page 1354).

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 Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 566: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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 "2022-09-01T00:00:00Z" 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 567: 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 array containing the metadata fields populated for the certificate request.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.30.4 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 1259 and Workflow Instances on page 1354).

Tip: The following permissions (see Security Overview) are required to use this feature:
WorkflowManagement: *Read*

Table 568: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Certificate Search Page*. 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 "2022-09-01T00:00:00Z" 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 569: 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 array containing the metadata fields populated for the certificate request.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.30.5 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. The certificate request IDs should be supplied in the request body as a JSON array of integers. 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 1259 and Workflow Instances on page 1354).

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 Overview) are required to use this feature:
WorkflowManagement: *Participate*

Table 570: POST Workflow Certificates Deny Input Parameters

Name	In	Description
CertificateRequestIds	Body	**Required**. An array of 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 1249) 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 571: POST Workflow Certificates Deny Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Successes</td><td>An array of 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 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.30.6 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. The certificate request IDs should be supplied in the request body as a JSON array of integers. 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 page 1259 and Workflow Instances on page 1354).

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 Overview) are required to use this feature:
WorkflowManagement: *Participate*

Table 572: POST Workflow Certificates Approve Input Parameters

Name	In	Description
requestIds	Body	**Required**. An array of 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 1249) to retrieve a list of all the certificate requests to determine the certificate request's IDs.

Table 573: POST Workflow Certificates Approve Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Successes</td><td>An array of 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 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 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31 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) 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.

For more information about workflows, see *Workflow Definitions* in the *Keyfactor Command Reference Guide*.

Table 574: 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 1263
/{definitionId}	GET	Returns details of the workflow definition, including steps, for the workflow with the specified GUID.	GET Workflow Definitions Definition ID on page 1263
/{definitionId}	PUT	Updates the name and description of the workflow definition with the specified GUID.	PUT Workflow Definitions Definition ID on page 1280
/	GET	Returns a list of workflow definitions, without steps.	GET Workflow Definitions on page 1297
/	POST	Creates a new workflow definition, without steps.	POST Workflow Definitions on page 1299
/Steps	GET	Returns information about the structure of the workflow definitions.	GET Workflow Definitions Steps on page 1316
/Types	GET	Returns a list of the defined workflow definition types.	GET Workflow Definitions Types on page 1318
/{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 1319
/{definitionId}/Publish	POST	Publishes the workflow definition with the specified GUID to activate it for use.	POST Workflow Definitions Definition ID Publish on page 1338

2.2.31.1 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 Overview) are required to use this feature:
WorkflowDefinitions: *Read*

Table 575: 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 1316) to retrieve a list of all the workflow definition steps to determine the extensionName.

Table 576: 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 *RequireApproval* step. • **EnrollStep**—Enroll for a certificate through Keyfactor Command. • **NOOPStep**—An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. • **PowerShell**—Run a PowerShell script. The script contents are embedded within the step. It does not call out to an external file. • **RequireApproval**—Require approval for a workflow step before the step can be completed. This step includes logic to gather the correct number of approvals from the users with the correct security roles and to send an email message indicating whether the step was approved or denied. This step does not include logic to send an email initiating the approval process. Use an *Email* type 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*. • **RestRequest**—Run a REST request. The REST request contents are embedded within the step. It does not call out to an external file. • **RevokeStep**—Revoke a certificate through Keyfactor Command.
Outputs	An object 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.2 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 Overview) are required to use this feature:
WorkflowDefinitions: *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 577: 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 1297) to retrieve a list of all the workflow definitions to determine the GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.3 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 Overview) are required to use this feature:
WorkflowDefinitions: *Read*

Table 578: 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 1297) 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 908) 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 579: 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.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template.
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: • Enrollment • Revocation
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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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: • Where-Object • ForEach-Object • Get-Command • **CustomPowerShell** Run a PowerShell script. The script contents are in a file placed in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory for Keyfactor Command. By default, this is: C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1). • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step applies to certificate enrollments, renewals, and revocations and 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*.</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 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. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent 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 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 </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>formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as Build from this Active Directory information must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step.

  Important: The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the Keyfactor Windows Enrollment Gateway Installation and Configuration Guide.

• EnrollStep
Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step.
• NOOPStep
An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow.
• RevokeStep
Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately following the EndNOOP step.

  Tip: For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 1098) 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>ConfigurationParameters</td><td>An array 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).</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Note: There are no ConfigurationParameters for steps of type *SubjectFormatter*, *EnrollStep*, *NOOPStep*, or *RevokeStep*.

Possible **CustomPowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script.
ScriptName	The path and filename for the script to execute. The script needs to be in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory. By default, this is: `C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow` The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1).

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

Name	Description
	Name / **Description**

Value	Description
	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: $(request:dn)</td><td>App Owner Last Name: $(metadata:AppOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:AppOwnerEmailAddress)</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 1: Tokens for Workflow Definitions in the *Keyfactor Command Reference Guide* for a complete list of available tokens.
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. • 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 $(re-

Name	Description

Name	Description

quester) 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 **PowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append 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

Name	Description
	Name / **Description**

Value	Description
	denied.
DenialEmailMessage	A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: 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. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).
ApprovalEmailSubject	A string indicating the subject line for the email message that will be delivered if the request is approved.
ApprovalEmailMessage	A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* for a

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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. • 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 **RestRequest** parameters include:

Value	Description
Headers	An array of key/value pair strings 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:

Name	Description
	Name / **Description** **Value** / **Description** `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration. DataBucketProperty — A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)` Verb — A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS

Name	Description
	Name / **Description** **Value** / **Description** • POST • PUT • TRACE UseBasicAuth: A Boolean indicating whether Basic authentication should be used for the request (True) or not (False). If *UseBasicAuth* is *False*, Windows authentication in the context of the Keyfactor Command application pool user will be used (see *Create Active Directory Service Accounts for Keyfactor Command* in the *Keyfactor Command Server Installation Guide*). BasicUsername: An array indicating the username information to use for authentication if *UseBasicAuth* is *True*. Supported methods to store credential 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 *PAM Providers* and *Privileged Access Management (PAM)* in the *Keyfactor Command Reference Guide* for more information. **Value** / **Description** SecretValue: A string containing the username defined for basic authentication (in DOMAIN\\username format). Parameters: An array indicating the parameters to supply for PAM authentication. These will vary depending on the PAM provider.

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Value / **Description**

Value	Description
Provider	A string indicating the ID of the PAM provider. Use the *GET /PamProviders* method (see GET PAM Providers on page 739) to retrieve a list of all the PAM providers to determine the ID.

For example, the username stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "KEYEXAMPLE\svc_MyServiceName"
}
```

The username 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 739 and the Folder and Object reference the folder name and object name in the CyberArk safe):

```
{
   "Provider": "1",
   "Parameters":{
      "Folder":"MyFolderName",
      "Object":"MyWorkflowUsername"
   }
}
```

The username 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 739 and the SecretId is the ID if the secret created in the Delinea secret server for this purpose):

Name	Description
	Name / **Description**

Name

Description

Value	Description
	<pre>{ "Provider": "1", "Parameters":{ "SecretId":"MyUsernameId" } }</pre> Due to its sensitive nature, this value is not returned in responses.
BasicPassword	An array 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)` **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`

<table>
<tr><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>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>ContentType</td><td>A string indicating the content type for the request. Supported values are:
application/json</td></tr>
<tr><td>RequestContent</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 ID 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>
<tr><td>Outputs</td><td>An array 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>
</table>

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. For a newly created workflow, this value will be null.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.4 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 Overview) are required to use this feature:
WorkflowDefinitions: *Modify*

Note: Only one workflow definition can be created for each combination of **Workflow Type** and **Key** (**Template**). In other words, you cannot have two enrollment or revocation workflow definitions for the same template, though you can have one enrollment workflow definition and one revocation workflow definition for a given template.

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.

Warning: 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 580: 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 581: PUT Workflow Definitions {definitionid} Response Body

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.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template.
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: • Enrollment • Revocation
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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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: • Where-Object • ForEach-Object • Get-Command • **CustomPowerShell** Run a PowerShell script. The script contents are in a file placed in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory for Keyfactor Command. By default, this is: C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1). • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step applies to certificate enrollments, renewals, and revocations and 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*.</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 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. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent 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 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 </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>formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as Build from this Active Directory information must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step.

  Important: The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the Keyfactor Windows Enrollment Gateway Installation and Configuration Guide.

• EnrollStep
Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step.
• NOOPStep
An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow.
• RevokeStep
Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately following the EndNOOP step.

  Tip: For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 1098) 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>ConfigurationParameters</td><td>An array 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).</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Description

Note: There are no ConfigurationParameters for steps of type *SubjectFormatter*, *EnrollStep*, *NOOPStep*, or *RevokeStep*.

Possible **CustomPowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script.
ScriptName	The path and filename for the script to execute. The script needs to be in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory. By default, this is: `C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow` The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1).

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

Name	Description
	Name / **Description**

Value	Description
	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: $(request:dn)</td><td>App Owner Last Name: $(metadata:AppOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:AppOwnerEmailAddress)</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 1: Tokens for Workflow Definitions in the *Keyfactor Command Reference Guide* for a complete list of available tokens.
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. • 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 $(re-

Name	Description
	Name / **Description**

quester) 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 **PowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append 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

Name	Description
	Name / **Description**

Value	Description
	denied.
DenialEmailMessage	A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: 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. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).
ApprovalEmailSubject	A string indicating the subject line for the email message that will be delivered if the request is approved.
ApprovalEmailMessage	A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* for a

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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. • 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 **RestRequest** parameters include:

Value	Description
Headers	An array of key/value pair strings 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:

Name	Description
	Name / **Description** **Value** / **Description** `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration. DataBucketProperty — A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)` Verb — A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS

<table>
<tr><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>

POST
PUT
TRACE

</td></tr>
<tr><td>UseBasicAuth</td><td>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).</td></tr>
<tr><td>BasicUsername</td><td>An array indicating the username information to use for authentication if UseBasicAuth is True.
Supported methods to store credential 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 PAM Providers and 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 username defined for basic authentication (in DOMAIN\\username format).</td></tr>
<tr><td>Parameters</td><td>An array 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>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Value / **Description**

Value	Description
Provider	A string indicating the ID of the PAM provider. Use the *GET /PamProviders* method (see GET PAM Providers on page 739) to retrieve a list of all the PAM providers to determine the ID.

For example, the username stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "KEYEXAMPLE\svc_MyServiceName"
}
```

The username 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 739 and the Folder and Object reference the folder name and object name in the CyberArk safe):

```
{
   "Provider": "1",
   "Parameters":{
      "Folder":"MyFolderName",
      "Object":"MyWorkflowUsername"
   }
}
```

The username 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 739 and the SecretId is the ID if the secret created in the Delinea secret server for this purpose):

Name	Description
	Name / **Description**

Name

Description

Value	Description
	<pre>{ "Provider": "1", "Parameters":{ "SecretId":"MyUsernameId" } }</pre> Due to its sensitive nature, this value is not returned in responses.
BasicPassword	An array 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)` **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`

<table>
<tr><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>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>ContentType</td><td>A string indicating the content type for the request. Supported values are:
application/json</td></tr>
<tr><td>RequestContent</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 ID 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>
<tr><td>Outputs</td><td>An array 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>
</table>

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. For a newly created workflow, this value will be null.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.5 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 1263) to return details including the workflow steps.

Tip: The following permissions (see Security Overview) are required to use this feature:
WorkflowDefinitions: *Read*

Table 582: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Workflow Definitions Search Feature*. The query fields supported for this endpoint are: • *DisplayName* • *Id* • *IsPublished* (true or false) • *WorkflowType* (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 583: 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.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template.
WorkflowType	A string indicating the type of workflow definition. The currently supported types are: • Enrollment • Revocation
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.6 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 1319). This method returns HTTP 200 OK on a success with details about the workflow definition.

Tip: The following permissions (see Security Overview) are required to use this feature:
WorkflowDefinitions: *Modify*

Note: Only one workflow definition can be created for each combination of **Workflow Type** and **Key** (**Template**). In other words, you cannot have two enrollment or revocation workflow definitions for the same template, though you can have one enrollment workflow definition and one revocation workflow definition for a given template.

Table 584: 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. Use the GET /Templates method (see GET Templates on page 1206) to retrieve a list or your certificate templates to determine the template 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.
WorkflowType	Body	**Required**. A string indicating the type of workflow definition. The currently supported types are: • Enrollment • Revocation This field cannot be modified on an edit.

Table 585: POST Workflow Definitions Response Body

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.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template.
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: • Enrollment • Revocation
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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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: • Where-Object • ForEach-Object • Get-Command • **CustomPowerShell** Run a PowerShell script. The script contents are in a file placed in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory for Keyfactor Command. By default, this is: C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1). • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step applies to certificate enrollments, renewals, and revocations and 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*.</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 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. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent 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 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 </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as *Build from this Active Directory information* must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. **Important:** The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the *Keyfactor Windows Enrollment Gateway Installation and Configuration Guide*. • **EnrollStep** Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. • **NOOPStep** An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. • **RevokeStep** Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately following the EndNOOP step. **Tip:** For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 1098) 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>ConfigurationParameters</td><td>An array 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*).</td></tr></table>

Name	Description

Name	Description

Note: There are no ConfigurationParameters for steps of type *SubjectFormatter*, *EnrollStep*, *NOOPStep*, or *RevokeStep*.

Possible **CustomPowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script.
ScriptName	The path and filename for the script to execute. The script needs to be in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory. By default, this is: `C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow` The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1).

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

Name	Description
	Name / **Description**

Value	Description
	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: $(request:dn)</td><td>App Owner Last Name: $(metadata:AppOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:AppOwnerEmailAddress)</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 1: Tokens for Workflow Definitions in the *Keyfactor Command Reference Guide* for a complete list of available tokens.
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. • 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 $(re-

Name	Description
	Name / **Description**

quester) 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 **PowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append 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

Name	Description
	Name / **Description**

Value	Description
	denied.
DenialEmailMessage	A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: 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. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).
ApprovalEmailSubject	A string indicating the subject line for the email message that will be delivered if the request is approved.
ApprovalEmailMessage	A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* for a

Name	Description
	Name / **Description**

Name

Description

Value	Description
	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. • 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 **RestRequest** parameters include:

Value	Description
Headers	An array of key/value pair strings 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:

Name	Description
	Name / **Description**

Name

Description

Value	Description
	<pre>"Headers": { "x-keyfactor-requested-with": ["APIClient"], "x-keyfactor-api-version": ["2"] }</pre> **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration.
DataBucketProperty	A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)`
Verb	A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS

<table>
<tr><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>

POST
PUT
TRACE

</td></tr>
<tr><td>UseBasicAuth</td><td>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).</td></tr>
<tr><td>BasicUsername</td><td>An array indicating the username information to use for authentication if UseBasicAuth is True.
Supported methods to store credential 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 PAM Providers and 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 username defined for basic authentication (in DOMAIN\\username format).</td></tr>
<tr><td>Parameters</td><td>An array 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>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Value / **Description**

Value	Description
Provider	A string indicating the ID of the PAM provider. Use the *GET /PamProviders* method (see GET PAM Providers on page 739) to retrieve a list of all the PAM providers to determine the ID.

For example, the username stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "KEYEXAMPLE\svc_MyServiceName"
}
```

The username 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 739 and the Folder and Object reference the folder name and object name in the CyberArk safe):

```
{
   "Provider": "1",
   "Parameters":{
      "Folder":"MyFolderName",
      "Object":"MyWorkflowUsername"
   }
}
```

The username 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 739 and the SecretId is the ID if the secret created in the Delinea secret server for this purpose):

Name	Description
	Name / **Description**

Name

Description

Value	Description
	<pre>{ "Provider": "1", "Parameters":{ "SecretId":"MyUsernameId" } }</pre> Due to its sensitive nature, this value is not returned in responses.
BasicPassword	An array 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)` **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`

<table>
<tr><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>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>ContentType</td><td>A string indicating the content type for the request. Supported values are:
application/json</td></tr>
<tr><td>RequestContent</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 ID 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>
<tr><td>Outputs</td><td>An array 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>
</table>

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. For a newly created workflow, this value will be null.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.7 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 Overview) are required to use this feature:
WorkflowDefinitions: *Read*

Table 586: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Workflow Definitions Search Feature*. The query fields supported for this endpoint are *DisplayName*, *ExtensionName*, and *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 587: 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 *RequireApproval* step. • **EnrollStep**—Enroll for a certificate through Keyfactor Command. • **NOOPStep**—An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. • **PowerShell**—Run a PowerShell script. The script contents are embedded within the step. It does not call out to an external file. • **RequireApproval**—Require approval for a workflow step before the step can be completed. This step includes logic to gather the correct number of approvals from the users with the correct security roles and to send an email message indicating whether the step was approved or denied. This step does not include logic to send an email initiating the approval process. Use an *Email* type 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*. • **RestRequest**—Run a REST request. The REST request contents are embedded within the step. It does not call out to an external file. • **RevokeStep**—Revoke a certificate through Keyfactor Command.
SupportedWorkflowTypes	An array containing a list of the workflow types supported by the workflow definition step. Possible built-in values are: • Enrollment • Revocation
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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.8 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 Overview) are required to use this feature:
WorkflowDefinitions: *Read*

Table 588: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Workflow Definitions Search Feature*. The query field supported for this endpoint is *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 589: GET Workflow Definitions Types Response Data

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>WorkflowType</td><td>A string indicating the display name of the workflow type.</td></tr>
<tr><td>KeyType</td><td>A string indicating the key type for the workflow. The built-in enrollment and revocation workflows use Templates as the key type.</td></tr>
<tr><td>ContextParameters</td><td>An object containing the tokens that the workflow type provider has the ability to replace. These will vary depending on the workflow type.</td></tr>
<tr><td>BuiltInSteps</td><td>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:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>DisplayName</td><td>A string indicating the display name for the step.</td></tr>
<tr><td>ExtensionName</td><td>A string indicating the extension name for the step. The built-in extensions are:
EnrollStep
RevokeStep</td></tr>
<tr><td>Outputs</td><td>An array 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.</td></tr>
<tr><td>ConfigurationParametersDefinition</td><td>An array containing the configuration parameters for the workflow definition step. These will vary depending on the step.</td></tr>
<tr><td>SignalsDefinition</td><td>An array containing the signals defined for the workflow definition step. These will vary depending on the step.</td></tr>
</table></td></tr>
</table>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.9 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 Overview) are required to use this feature:
WorkflowDefinitions: *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.

Warning: 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 590: 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 1297) to retrieve a list of all the workflow definitions to determine the GUID.

<table>
<tr><th>Name</th><th>In</th><th>Description</th></tr>
<tr><td>request</td><td>Body</td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>DisplayName</td><td>A string indicating the display name for the step.</td></tr>
<tr><td>UniqueName</td><td>A string indicating the unique name for the step. This value must be unique among the steps in the particular workflow definition. It is intended to be used as a user-friendly reference ID.</td></tr>
<tr><td>ExtensionName</td><td>A string indicating the type of step. The currently supported types are:
Email
Send an email message. This is a separate email message from those typically sent as part of a Require Approval step. You might send an email message as part of an enrollment request to notify approvers that a new request needs approval. The email messages can be customized to provide detailed information about, for example, the certificate request.
PowerShell
Run PowerShell commands within the confines of the workflow to populate variables with information to pass back to the workflow. The PowerShell script contents are embedded within the step. This step does not call out to an external file. This provides a high level of security by greatly limiting the number of standard PowerShell cmdlets that can be executed by the workflow step. A small number of PowerShell cmdlets have been white listed to allow them to be included in workflow steps of this type, including:
Where-Object
ForEach-Object
Get-Command
CustomPowerShell
Run a PowerShell script. The script contents are in a file placed in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory for Keyfactor Command. By default, this is:
<code>C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow</code>
The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1).
RequireApproval
Require approval for a workflow step before the step can be completed. The require approval step applies to certificate enrollments, renewals, and revocations and 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</td></tr>
</table>
</td></tr>
</table>

Table 591: PUT Workflow Definitions {definitionid} Steps Response Body

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.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template.
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: • Enrollment • Revocation
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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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: • Where-Object • ForEach-Object • Get-Command • **CustomPowerShell** Run a PowerShell script. The script contents are in a file placed in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory for Keyfactor Command. By default, this is: C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1). • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step applies to certificate enrollments, renewals, and revocations and 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*.</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 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. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent 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 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 </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as *Build from this Active Directory information* must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. **Important:** The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the *Keyfactor Windows Enrollment Gateway Installation and Configuration Guide*. • **EnrollStep** Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. • **NOOPStep** An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. • **RevokeStep** Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately following the EndNOOP step. **Tip:** For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 1098) 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>ConfigurationParameters</td><td>An array 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*).</td></tr></table>

Name	Description
	Name / **Description**

Note: There are no ConfigurationParameters for steps of type *SubjectFormatter*, *EnrollStep*, *NOOPStep*, or *RevokeStep*.

Possible **CustomPowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script.
ScriptName	The path and filename for the script to execute. The script needs to be in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory. By default, this is: `C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow` The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1).

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

Name	Description
	Name / **Description**

Value	Description
	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: $(request:dn)</td><td>App Owner Last Name: $(metadata:AppOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:AppOwnerEmailAddress)</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 1: Tokens for Workflow Definitions in the *Keyfactor Command Reference Guide* for a complete list of available tokens.
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. • 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 $(re-

Name	Description

Name	Description
	quester) 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 **PowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the *value*.
ScriptContent	A string containing the PowerShell commands to execute. This should be the actual contents of the PowerShell script (the PowerShell commands and supporting components), not a path and filename to an external file.

Tip: Tokens (a.k.a. substitutable special text) may be used in the script parameter value field. Tokens use a variable in the workflow definition that is replaced by data from the certificate request, certificate, or certificate metadata at processing time. For example, you can take the revocation comment entered when the revocation request is approved—$(cmnt)—and append 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

Name	Description
	Name / **Description**

Value	Description
	denied.
DenialEmailMessage	A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: 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. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).
ApprovalEmailSubject	A string indicating the subject line for the email message that will be delivered if the request is approved.
ApprovalEmailMessage	A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* for a

Name	Description
	Name / **Description**

Value	Description
	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. • 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 **RestRequest** parameters include:

Value	Description
Headers	An array of key/value pair strings 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:

Name	Description
	Name / **Description** **Value** / **Description** `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration. DataBucketProperty — A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)` Verb — A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS

<table>
<tr><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>

POST
PUT
TRACE

</td></tr>
<tr><td>UseBasicAuth</td><td>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).</td></tr>
<tr><td>BasicUsername</td><td>An array indicating the username information to use for authentication if UseBasicAuth is True.
Supported methods to store credential 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 PAM Providers and 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 username defined for basic authentication (in DOMAIN\\username format).</td></tr>
<tr><td>Parameters</td><td>An array 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>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Value / **Description**

Value	Description
Provider	A string indicating the ID of the PAM provider. Use the *GET /PamProviders* method (see GET PAM Providers on page 739) to retrieve a list of all the PAM providers to determine the ID.

For example, the username stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "KEYEXAMPLE\svc_MyServiceName"
}
```

The username 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 739 and the Folder and Object reference the folder name and object name in the CyberArk safe):

```
{
   "Provider": "1",
   "Parameters":{
      "Folder":"MyFolderName",
      "Object":"MyWorkflowUsername"
   }
}
```

The username 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 739 and the SecretId is the ID if the secret created in the Delinea secret server for this purpose):

Name	Description
	Name / **Description**

Name

Description

Value	Description
	<pre>{ "Provider": "1", "Parameters":{ "SecretId":"MyUsernameId" } }</pre> Due to its sensitive nature, this value is not returned in responses.
BasicPassword	An array 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)` **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`

<table>
<tr><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>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>ContentType</td><td>A string indicating the content type for the request. Supported values are:
application/json</td></tr>
<tr><td>RequestContent</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 ID 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>
<tr><td>Outputs</td><td>An array 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>
</table>

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. For a newly created workflow, this value will be null.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.31.10 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 1354) 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 Overview) are required to use this feature:
WorkflowDefinitions: *Modify*

Table 592: 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 1297) to retrieve a list of all the workflow definitions to determine the GUID.

Table 593: POST Workflow Definitions {definitionid} Publish Response Body

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.
KeyDisplayName	A string indicating the friendly name defined in Keyfactor Command for the certificate template.
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: • Enrollment • Revocation
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

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>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: • Where-Object • ForEach-Object • Get-Command • **CustomPowerShell** Run a PowerShell script. The script contents are in a file placed in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory for Keyfactor Command. By default, this is: C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1). • **RequireApproval** Require approval for a workflow step before the step can be completed. The require approval step applies to certificate enrollments, renewals, and revocations and 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*.</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 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. The REST request contents are embedded within the step. It does not call out to an external file. - EnrollmentAgent 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 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 </td></tr></table>

Name	Description
	<table><tr><th>Name</th><th>Description</th></tr><tr><td></td><td>formatting the incoming subject, SANs, and/or SID in the certificate request appropriately such that the enrollment will complete successfully with the target CA and Keyfactor Command template, which is not configured to build from AD. Any Keyfactor Windows Enrollment Gateway using a client-side template configured with the subject as *Build from this Active Directory information* must be configured with a workflow step of this type on the Keyfactor Command template that has been mapped in the gateway to that template in order to complete an enrollment through the gateway. There are no configuration parameters for the step. **Important:** The template in Keyfactor Command that is mapped to the client-side template configured to build the subject from Active Directory also needs to be configured with three enrollment fields to support handling the incoming subject, SANs, and/or SID. For more information about configuring this, see the *Keyfactor Windows Enrollment Gateway Installation and Configuration Guide*. • **EnrollStep** Enroll for a certificate through Keyfactor Command. The enroll step must always fall as the last step in the workflow, immediately following the EndNOOP step. • **NOOPStep** An entry or exit step in which no operation occurs. Steps of this type indicate the start and end of the workflow. • **RevokeStep** Revoke a certificate through Keyfactor Command. The revoke step must always fall as the last step in the workflow, immediately following the EndNOOP step. **Tip:** For steps that send email messages, the SMTP settings and sender information come from the standard Keyfactor Command SMTP configuration (see SMTP on page 1098) 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>ConfigurationParameters</td><td>An array 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*).</td></tr></table>

Name	Description

Name	Description

Note: There are no ConfigurationParameters for steps of type *SubjectFormatter*, *EnrollStep*, *NOOPStep*, or *RevokeStep*.

Possible **CustomPowerShell** parameters include:

Value	Description
ScriptParameters	An array of key/value pair strings defining any parameters to be used in the PowerShell script.
ScriptName	The path and filename for the script to execute. The script needs to be in the ExtensionLibrary\Workflow directory or a subdirectory of it on the Keyfactor Command server under the install directory. By default, this is: `C:\Program Files\Keyfactor\Keyfactor Platform\ExtensionLibrary\Workflow` The file must have an extension of .ps1. A sample PowerShell script is provided in the Workflow directory (CustomPowershellExample.ps1).

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

Name	Description
	Name / **Description**

Value	Description
	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: $(request:dn)</td><td>App Owner Last Name: $(metadata:AppOwnerLastName)</td></tr>\n<tr><td>SANs: $(sans)</td><td>App Owner Email Address: $(metadata:AppOwnerEmailAddress)</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 1: Tokens for Workflow Definitions in the *Keyfactor Command Reference Guide* for a complete list of available tokens.
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. • 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 $(re-

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
quester) 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 PowerShell parameters include:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>ScriptParameters</td><td>An array of key/value pair strings defining any parameters to be used in the PowerShell script. The key is the name of a custom parameter defined by you and the value is the initial value that should be set for that parameter before the PowerShell is executed, if any. Tokens are supported in the value.</td></tr>
<tr><td>ScriptContent</td><td>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.</td></tr>
</table>

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:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>MinimumApprovals</td><td>In integer indicating the minimum number of users who must approve the request to allow the request to complete.</td></tr>
<tr><td>DenialEmailSubject</td><td>A string indicating the subject line for the email message that will be delivered if the request is</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Name	Description
	Name / **Description**

Value	Description
	denied.
DenialEmailMessage	A string indicating the email message that will be delivered if the request is denied. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: 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. • Your custom email-based metadata field, which would be specified similarly to $(metadata:AppOwnerEmailAddress).
ApprovalEmailSubject	A string indicating the subject line for the email message that will be delivered if the request is approved.
ApprovalEmailMessage	A string indicating the email message that will be delivered if the request is approved. The email message is made up of regular text and tokens. If desired, you can format the message body using HTML. See *Table: Tokens for Workflow Definitions* in the *Keyfactor Command Reference Guide* for a

Name	Description
	Name / **Description**

Value	Description
	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. • 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 **RestRequest** parameters include:

Value	Description
Headers	An array of key/value pair strings 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:

Name	Description
	Name / **Description** **Value** / **Description** `"Headers": { "x-keyfactor-requested-with": [ "APIClient" ], "x-keyfactor-api-version": [ "2" ] }` **Tip:** For a Keyfactor API request, version 1 is assumed if no version is specified. Content type and authorization headers do not need to be specified, since those are addressed elsewhere in the configuration. DataBucketProperty — A string containing the variable that the response from the request will be returned in, if any. You can then reference this parameter from subsequent steps in the workflow. **Tip:** The response is stored as a serialized JObject. To make use of only a portion of the response data in your subsequent step, use JSON path syntax. For example, say you returned the data from a GET /Agents request in a variable called *MyResponse* and you wanted to reference the *ClientMachine* name for the orchestrator in a subsequent email message. To limit the data to the first result and only the ClientMachine name, in the email message you would enter the following: `$(MyResponse.[0].ClientMachine)` Verb — A string indicating the HTTP verb for the type of request to perform. Supported values are: • DELETE • GET • HEAD • OPTIONS

<table>
<tr><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>

POST
PUT
TRACE

</td></tr>
<tr><td>UseBasicAuth</td><td>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).</td></tr>
<tr><td>BasicUsername</td><td>An array indicating the username information to use for authentication if UseBasicAuth is True.
Supported methods to store credential 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 PAM Providers and 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 username defined for basic authentication (in DOMAIN\\username format).</td></tr>
<tr><td>Parameters</td><td>An array 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>
</td></tr>
</table>

Name	Description
	Name / **Description**

Name

Description

Value	Description
	Value / **Description**

Value	Description
Provider	A string indicating the ID of the PAM provider. Use the *GET /PamProviders* method (see GET PAM Providers on page 739) to retrieve a list of all the PAM providers to determine the ID.

For example, the username stored as a Keyfactor secret will look like:

```
{
   "SecretValue": "KEYEXAMPLE\svc_MyServiceName"
}
```

The username 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 739 and the Folder and Object reference the folder name and object name in the CyberArk safe):

```
{
   "Provider": "1",
   "Parameters":{
      "Folder":"MyFolderName",
      "Object":"MyWorkflowUsername"
   }
}
```

The username 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 739 and the SecretId is the ID if the secret created in the Delinea secret server for this purpose):

Name	Description
	Name / **Description**

Name

Description

Value	Description
	<pre>{ "Provider": "1", "Parameters":{ "SecretId":"MyUsernameId" } }</pre> Due to its sensitive nature, this value is not returned in responses.
BasicPassword	An array 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)` **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`

<table>
<tr><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>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>ContentType</td><td>A string indicating the content type for the request. Supported values are:
application/json</td></tr>
<tr><td>RequestContent</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 ID 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>
<tr><td>Outputs</td><td>An array 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>
</table>

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. For a newly created workflow, this value will be null.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon () at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.32 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 594: 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 the next page
/	GET	Retrieve a list of the workflow instances.	GET Workflow Instances on page 1376
/My	GET	Retrieve the workflow instances created by the user making the API request.	GET Workflow Instances My on page 1379
/AssignedToMe	GET	Retrieve the workflow instances assigned to the user making the API request.	GET Workflow Instances AssignedToMe on page 1382
/{instanceId}/Stop	POST	Rejects a workflow instance, preventing it from continuing.	POST Workflow Instances Instance Id Stop on page 1386
/{instanceId}/Signals	POST	Input data to the workflow instance with the specified GUID.	POST Workflow Instances Instance ID Signals on page 1386
/{instanceId}/Restart	POST	Restart the specified workflow instance after a failure.	POST Workflow Instances Instance Id Restart on page 1389

2.2.32.1 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 Overview) are required to use this feature:
WorkflowInstances: *Manage*

Table 595: 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 1376) to retrieve a list of all the workflow instances to determine the GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.32.2 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 Overview) are required to use this feature:
WorkflowInstances: *ReadAll* OR
WorkflowInstances: *ReadAssignedToMe* OR
WorkflowInstances: *ReadMy*

Users with *ReadMy* or *ReadAssignedToMe* will only be able to retrieve the workflow instances created by them (*ReadMy*) or assigned to them (*ReadAssignedToMe*) unless they also have *ReadAll*.

Table 596: 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 1376) to retrieve a list of all the workflow instances to determine the GUID.

Table 597: 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 object 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*. StepSignalId — A string indicating the Keyfactor Command reference GUID of the signal in

<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>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 array 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:
• Enrollment
• Revocation</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 (DOMAIN\\username) followed by an indication of the type of action and a specific message about the action. For example:
"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=appssrvr14.keyexample.com."
Or
"KEYEXAMPLE\\jsmith is revoking certificate with CN=appsrvr12.keyexample.com."</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>

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>StartDate</td><td>A string indicating the date and time when the instance was initiated.</td></tr>
<tr><td>InitialData</td><td>An array containing the data included in the workflow instance when the workflow was initiated. Initial workflow instance data includes:
<table>
<tr><th>Name</th><th>Operation Type</th><th>Description</th></tr>
<tr><td>CertificateAuthority</td><td>Enrollment and Revocation</td><td>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.</td></tr>
<tr><td>CertificateId</td><td>Revocation</td><td>An integer indicating the Keyfactor Command reference ID for the certificate.</td></tr>
<tr><td>SerialNumberString</td><td>Revocation</td><td>A string indicating the serial number of the certificate being revoked.</td></tr>
<tr><td>Thumbprint</td><td>Revocation</td><td>A string indicating the thumbprint of the certificate being revoked.</td></tr>
<tr><td>RevokeCode</td><td>Revocation</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>
</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>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 certificate (true) or not (false).</td></tr>
<tr><td>OperationStart</td><td>Revoc-ation</td><td>A string indicating the time at which the revocation work-flow was initiated.</td></tr>
<tr><td>Template</td><td>Enroll-ment</td><td>A string indicating the certificate template short name used for the enrollment request.</td></tr>
<tr><td>IncludeChain</td><td>Enroll-ment</td><td>A Boolean indicating whether to include the certificate chain in the enrollment response (true) or not (false).</td></tr>
<tr><td>SANs</td><td>Enroll-ment</td><td>An array of key/value pairs indicating the subject alter-native names (SANs) for the certificate requested in the enrollment. 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>
</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>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>dns</td><td>DNS Name</td></tr>
<tr><td>directory</td><td>Directory Name</td></tr>
<tr><td>uri</td><td>Uniform Resource Identifier</td></tr>
<tr><td>ip4</td><td>IP v4 Address</td></tr>
<tr><td>ip6</td><td>IP v6 Address</td></tr>
<tr><td>registeredid</td><td>Registered ID (an OID)</td></tr>
<tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr>
<tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr>
</table>
For example:
<pre>
"SANs": {
   "dns": [
      "dnssan1.keyexample.com",
      "dnssan2.keyexample.com",
      "dnssan3.keyexample.com"
   ],
   "ip4": [
      "192.168.2.73"
   ]
}
</pre>
</td></tr>
<tr><td>AdditionalAttributes</td><td>Enroll-ment</td><td>An array of key/value pairs 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>Enroll-ment</td><td>An array of key/value pairs 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</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>and the value is the value for the field.</td></tr>
<tr><td>Format</td><td>Enroll-ment</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>Enroll-ment</td><td>A string indicating a custom friendly name for the certificate.</td></tr>
<tr><td>Subject</td><td>Enroll-ment</td><td>A string containing the subject name of the requested certificate using X.500 format.</td></tr>
<tr><td>RenewalCertificate</td><td>Enroll-ment</td><td>An array containing the certificate information for the certificate that is being renewed. Certificate data includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Certificate</td><td>An array containing a key value pair 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 639).</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>Stores</td><td>Enroll-ment</td><td>An object containing a comma delimited set of arrays 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>An array of GUIDs indicating the certificate store(s) to which the certificate should be deployed.
Use the GET /CertificateStores method (see GET Certificate Stores on page 384) 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>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 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 224) 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>Prop-erties</td><td>An array of key/value pairs for the unique</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>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>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:
<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>
</td></tr>
</table>
</td></tr>
</table>

Name	Description

Name	Operation Type	Description
ManagementJobTime	Enrollment	An array indicating the schedule for the management job to add the certificate to the 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). **Tip:** In some instances, jobs initially scheduled as *Immediate* will appear on a GET as *null*.
ExactlyOnce	A dictionary that indicates a job scheduled to run at the time specified with the parameter:

Name	Description
Time	The date and time to next run the job. The date and time should be given using the ISO 8601 UTC time format YYYY-MM-DDTHH:mm:ss.000Z (e.g. 2021-05-19T16:23:01Z).

For example, exactly once at 11:45 am:

```
"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}
```

<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 ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>IsPFX</td><td>Enroll-ment</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>PfxPass-wordSecretIn-stanceId</td><td>Enroll-ment</td><td>A string indicating the Keyfactor Command reference GUID for the PFX password used to secure the PFX file on down-load.</td></tr>
<tr><td>InitiatingUserName</td><td>Enroll-ment and Revoc-ation</td><td>A string indicating the name of the user who initiated the workflow in DOMAIN\\username format.</td></tr>
</table>
</td></tr>
<tr><td>CurrentStateData</td><td>An array 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:
<table>
<tr><th>Name</th><th>Oper-ation Type</th><th>Description</th></tr>
<tr><td>CertificateAuthority</td><td>Enroll-ment and Revoc-ation</td><td>A string indicating the certificate authority that will be used to enroll against, for enrollment requests, or that issued the certificate, for revocation requests.</td></tr>
<tr><td>CertificateId</td><td>Revoc-ation</td><td>For revocation requests only, 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>SerialNumberString</td><td>Revocation</td><td>A string indicating the serial number of the certificate being revoked.</td></tr>
<tr><td>Thumbprint</td><td>Revocation</td><td>A string indicating the thumbprint of the certificate being revoked.</td></tr>
<tr><td>RevokeCode</td><td>Revocation</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>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>Revoc-</td><td>A Boolean indicating whether delegation is enabled 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>Oper-ation Type</th><th>Description</th></tr>
<tr><td></td><td>ation</td><td>certificate authority that issued the certificate (true) or not (false).</td></tr>
<tr><td>OperationStart</td><td>Revoc-ation</td><td>A string indicating the time at which the revocation work-flow 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 array of key/value pairs indicating the subject altern-ative names (SANs) for the certificate requested in the enrollment. Possible values for the key are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>rfc822</td><td>RFC 822 Name</td></tr>
<tr><td>dns</td><td>DNS Name</td></tr>
<tr><td>directory</td><td>Directory Name</td></tr>
<tr><td>uri</td><td>Uniform Resource Identifier</td></tr>
<tr><td>ip4</td><td>IP v4 Address</td></tr>
<tr><td>ip6</td><td>IP v6 Address</td></tr>
<tr><td>registeredid</td><td>Registered ID (an OID)</td></tr>
<tr><td>ms_ntprincipalname</td><td>MS_NTPrincipalName (a string)</td></tr>
<tr><td>ms_ntdsreplication</td><td>MS_NTDSReplication (a GUID)</td></tr>
</table>
For example:</td></tr>
</table>
</td></tr>
</table>

<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><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 array of key/value pairs 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 array of key/value pairs 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 array containing the certificate information for the certificate that is being renewed. Certificate data includes:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Certificate</td><td>An array containing a key value</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>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>pair 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 639).
</td></tr>
<tr><td>Stores</td><td>Enroll-ment</td><td>An object containing a comma delimited set of arrays 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>An array of GUIDs indicating the certificate store(s) to which the certificate should be deployed.
Use the GET /CertificateStores method (see GET Certificate Stores on page 384) with a query of "Approved -eq true" to retrieve a list of all your approved certi-</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>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>ficate stores to determine the GUID(s) of the store(s).</td></tr>
<tr><td>Alias</td><td>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 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 224) 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>Prop-erties</td><td>An array of key/value pairs 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</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>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>returned by GET 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>
</td></tr>
<tr><td>Management-mentJobTime</td><td>Enroll-ment</td><td>An array indicating the schedule for the management job to add the certificate to any certificate store(s). Possible management job time values include:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>Immediate</td><td>A Boolean that indicates a job scheduled to run immediately (true) or not (false).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<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 Imme-diate will appear on a GET as null.</td></tr>
<tr><td>ExactlyOnc-e</td><td>A dictionary that indicates a job sched-uled 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. 2021-05-19T16:23:01Z).</td></tr>
</table>
For example, exactly once at 11:45 am:
<pre>"ExactlyOnce": {
   "Time": "2022-02-27T11:45:00Z"
}</pre>
Tip: In some instances, jobs initially scheduled as Imme-diate will appear on a GET as ExactlyOnce.</td></tr>
</table>
</td></tr>
<tr><td>IsPFX</td><td>Enroll-ment</td><td>A Boolean indicating whether the certificate enrollment type that initiated the workflow instance was PFX (true) or</td></tr>
</table>
</td></tr>
</table>

Name	Description
	<table><tr><th>Name</th><th>Operation Type</th><th>Description</th></tr><tr><td></td><td></td><td>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>Enrollment and Revocation</td><td>A string indicating the name of the user who initiated the workflow in DOMAIN\\username format.</td></tr><tr><td>KeyRetention</td><td>Enrollment</td><td>A Boolean indicating whether the private key for the certificate resulting from the enrollment will be retained in Keyfactor Command (true) or not (false).</td></tr><tr><td>CSR</td><td>Enrollment</td><td>A string containing the CSR generated for the certificate request.</td></tr><tr><td>(Custom)</td><td>Enrollment and Revocation</td><td>Optional user-generated custom fields returning response data from PowerShell scripts or REST requests.</td></tr><tr><td>CACertificate</td><td>Enrollment</td><td>An array containing the certificate information returned from the CA for the certificate that is being requested. CA certificate details include:<table><tr><th>Name</th><th>Description</th></tr><tr><td>CACertificateId</td><td>A string containing the ID assigned to the certificate by the CA.</td></tr><tr><td>CARequestID</td><td>A string containing the ID assigned to the certificate request by the CA.</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>Status</td><td>An integer indicating the status for the certificate as returned by the CA.</td></tr>
<tr><td>Certificate</td><td>A string containing the certificate as returned by the CA in base-64 encoded binary format.</td></tr>
<tr><td>CertificateTemplate</td><td>A string indicating the certi-ficate template used to issue the certificate.</td></tr>
<tr><td>RevocationDate</td><td>A string indicating the revocation date for the certificate as returned by the CA.</td></tr>
<tr><td>RevocationReason</td><td>A string indicating the revocation reason for the certificate as returned by the CA.</td></tr>
<tr><td>ArchivedKey</td><td>A Boolean indicating whether the certificate is configured for key archival on the CA (true) or not (false).</td></tr>
</table>
Note: This field is only populated only after the certificate has been issued by the CA.
</td></tr>
<tr><td>DispositionMessage</td><td>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</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>certificate request has been submitted to the CA.</td></tr>
<tr><td>CACer-tificateRequest</td><td>Enroll-ment</td><td>An array containing the certificate information for the certificate that is being requested. Certificate request data includes:
<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>
</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>KeyfactorId</td><td>Enroll-ment</td><td>An integer indicating the Keyfactor Command reference ID for the certificate.</td></tr>
<tr><td>KeyStatus</td><td>Enroll-ment</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>Priv-ateKeyConverter</td><td>Enroll-ment</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.32.3 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 Overview) are required to use this feature:
WorkflowInstances: *ReadAll*

Table 598: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Workflow Instances Search Feature*. 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 599: 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 array 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.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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:
Enrollment
Revocation</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 (DOMAIN\\username) followed by an indication of the type of action and a specific message about the action. For example:
"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=apps-srvr14.keyexample.com."
Or
"KEYEXAMPLE\\jsmith is revoking certificate with CN=apps-srvr12.keyexample.com."</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.32.4 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 Overview) are required to use this feature:
WorkflowInstances: *ReadAll* OR
WorkflowInstances: *ReadMy*

Table 600: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Workflow Instances Search Feature*. 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 601: 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 array 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.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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:
Enrollment
Revocation</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 (DOMAIN\\username) followed by an indication of the type of action and a specific message about the action. For example:
"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=apps-srvr14.keyexample.com."
Or
"KEYEXAMPLE\\jsmith is revoking certificate with CN=apps-srvr12.keyexample.com."</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.32.5 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 Overview) are required to use this feature:
WorkflowInstances: *ReadAll* OR
WorkflowInstances: *ReadAssignedToMe*

Table 602: 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 dropdowns for the same feature. For querying guidelines, refer to the *Keyfactor Command Reference Guide*: *Using the Workflow Instances Search Feature*. 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 603: 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 approval roles.* are returned using this method.
Definition	An array containing the workflow definition. Workflow definition data includes: **Name** / **Description** Id — A string indicating the Keyfactor Command reference GUID of the workflow definition.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Description</th></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:
Enrollment
Revocation</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 (DOMAIN\\username) followed by an indication of the type of action and a specific message about the action. For example:
"KEYEXAMPLE\\jsmith is enrolling for a certificate with CN=apps-srvr14.keyexample.com."
Or
"KEYEXAMPLE\\jsmith is revoking certificate with CN=apps-srvr12.keyexample.com."</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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.32.6 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 Overview) are required to use this feature:
WorkflowInstances: *Manage*

Table 604: 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 1376) to retrieve a list of all the workflow instances to determine the GUID.

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.32.7 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 Overview) 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 605: 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 GET Workflow Instances on page 1376) to retrieve a list of all the workflow instances to determine the GUID.</td></tr>
<tr><td>signal</td><td>Body</td><td>Required. An array 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. RequireApproval signal values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>SignalKey</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 GET Workflow Definitions Definition ID on page 1263) 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 GET Workflow Instances on page 1376, GET Workflow Instances AssignedToMe on page 1382, or GET Workflow Instances My on page 1379) to return the CurrentStepUniqueName.</td></tr>
<tr><td>Data</td><td>Required. An array containing key/value pairs 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></td></tr>
</table>
For example, to approve a Require Approval step called RequireApproval1 with a comment:
<pre>{</pre></td></tr>
</table>

Name	In	Description
		<pre>"SignalKey": "RequireApproval1.ApprovalStatus", "Data": { "Approved": "True", "Comment": "Here is my comment." } }</pre>

Tip: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.2.32.8 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 Overview) are required to use this feature:
WorkflowInstances: *Manage*
WorkflowDefinitions: *Read*

Table 606: 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 1376) 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: For code examples, see the *Keyfactor API Endpoint Utility*. To find the embedded web copy of this utility, click the help icon (?) at the top of the Keyfactor Command Management Portal page next to the **Log Out** button.

2.3 Classic API

The Keyfactor Classic API, also known as the CMS API, is the Web API that has been provided with Keyfactor Command for several product generations. The Classic API may be needed in your environment if you're upgrading and have written API applications using the Classic API. If you're new to building an API application to work with Keyfactor Command, Keyfactor strongly recommends that you use the newer Keyfactor API (see Keyfactor API on page 7).

2.3.1 Security Role Overview

In order to use the Classic API, certain security role permissions must be granted to the identity used by the client to authenticate to the API. Specifically, the user must have the *API Read* permission to make any requests. Beyond this, different API endpoints have different requirements (see Classic API Security Role Requirements below).

Where the table indicates that *Certificate Store Management* permissions are required, this can either be global permissions to all certificate stores or permissions granted to the specific certificate store using certificate store container security. Likewise, where *Certificates* permissions are required, this can either be global certificate permissions on all certificates or permissions granted to a specific certificate or set of certificates using certificate collection security. See the *Keyfactor Command Reference Guide* for more information about container and collection security.

Table 607: Classic API Security Role Requirements

Endpoint	Security Role Permissions
ApiApp/1/GetApiApps	System Settings: Read
ApiApp/1/AddApiApp	System Settings: Modify
ApiApp/1/EditApiApp	System Settings: Modify
ApiApp/1/DeleteApiApp	System Settings: Modify
CertEnroll/1/Pkcs10	None
CertEnroll/1/Pkcs12	None
CertEnroll/1/Templates	None
CertEnroll/1/Token	None
CertEnroll/2/Pkcs10	None

Endpoint	Security Role Permissions
CertEnroll/2/Pkcs12	None
CertEnroll/2/Templates	None
CertEnroll/2/Token	None
CertEnroll/3/Pkcs10	None
CertEnroll/3/Pkcs12	None
CertEnroll/3/Renew	Certificate Store Management: Read and Schedule
CertEnroll/3/Templates	None
Certificates/1/Metafield	Certificates: Modify and Certificate Metadata Types: Read
Certificates/2/Import	Certificates: Import
Certificates/3/Contents	Certificates: Read
Certificates/3/Count	Certificates: Read
Certificates/3/PublishCRL	PKI Management: Modify
Certificates/3/Recover	Certificates: Recover
Certificates/3/Revoke	Certificates: Revoke
Certificates/3/Search	Certificates: Read
Certstore/1/AddCert	Certificate Store Management: Read and Schedule, and Certificates: Read
Certstore/1/AddCertStore	Certificate Store Management: Modify
Certstore/1/AddCertStoreServer	Certificate Store Management: Modify
Certstore/1/AddPFX	Certificate Store Management: Read and Schedule
Certstore/1/CreateJKS	Certificate Store Management: Modify
Certstore/1/EditCertStore	Certificate Store Management: Modify
Certstore/1/EditCertStoreServer	Certificate Store Management: Modify
Certstore/1/Inventory	Certificate Store Management: Read
Certstore/1/Keystores	Certificate Store Management: Read

Endpoint	Security Role Permissions
Certstore/1/Remove	Certificate Store Management: Schedule and Certificates: Read
Certstore/1/ScheduleInventory	Certificate Store Management: Modify
Metadata/2/Compare	Certificates: Read and Certificate Metadata Types: Read
Metadata/2/Get	Certificates: Read and Certificate Metadata Types: Read
Metadata/2/Set	Certificates: Modify and Certificate Metadata Types: Read
Metadata/3/Get	Certificates: Read and Certificate Metadata Types: Read
Metadata/3/GetDefinition	Certificate Metadata Types: Read
Metadata/3/Set	Certificates: Modify and Certificate Metadata Types: Read
Security/1/GetIdentities	Security Settings: Read
Security/1/AddIdentity	Security Settings: Modify
Security/1/DeleteIdentity	Security Settings: Modify
Security/1/GetRoles	Security Settings: Read
Security/1/AddRole	Security Settings: Modify
Security/1/EditRole	Security Settings: Modify
Security/1/DeleteRole	Security Settings: Modify
SSL/1/AddEndpoint	SSL Management: Modify
SSL/1/AddEndpointGroup	SSL Management: Modify
SSL/1/Agents	SSL Management: Read
SSL/1/EndpointGroups	SSL Management: Read
Workflow/1/ApproveRequest	Workflow: Read and Participate
Workflow/1/DenyRequest	Workflow: Read and Participate

Endpoint	Security Role Permissions
Workflow/1/PendingList	Workflow: Read and Participate
Status	None
vSCEP	Configured through the Keyfactor Command Configuration Wizard or through the Application Settings page in the Keyfactor Command Management Portal.

2.3.2 ApiApp

The ApiApp component of the Keyfactor Web APIs includes all methods necessary to programmatically add, edit, get and delete API Applications. The complete set of endpoints is shown in2.3.2 ApiApp.

Table 608: ApiApp Endpoints

Endpoint	Method	Description
/1/GetApiApps	GET	Returns a list of the API applications
/1/AddApiApp	POST	Add an API application to Keyfactor Command
/1/EditApiApp	POST	Edit an API application in Keyfactor Command
/1/DeleteApiApp	POST	Deletes and API application from Keyfactor Command

2.3.2.1 ApiAPP GetApiApps

The GET GetApiApps endpoint returns a list of all API Applications defined in Keyfactor Command with the Id, Name, Key, Secret, CAId, CAConfiguration, TemplateId, TemplateName, TemplateForest and whether the Application is Enabled or not. No parameters or extra headers are required for this method.

Example Request

GET http://<host>/CMSApi/ApiApp/1/GetApiApps

Example Response

Status Code: 200

```
[
	{
		"Id": "<Id>",
		"Name": "<name>",
		"Key": "<hexadecimal key>",
```

```
            "Secret": "<hexadecimal secret>",
            “Enabled”: “True”,
            "CAId": "<CA Id>",
            "CAConfiguration": "<CA Host Name>\\<CA Logical Name>",
            "TemplateId": "<Template Id>",
            "TemplateName": "<Template Common Name>",
            "TemplateForest": "<Template Forest>"
        }
]
```

2.3.2.2 ApiApp AddApiApp

The POST AddApiApp endpoint adds an API Application to Keyfactor Command. It returns the Id of the newly added Application. Table 8 - AddApiApp Parameters shows the parameters that are used for the creation of API Applications through the Keyfactor Web APIs.

Table 609: AddApiApp Parameters

Parameter Name	Parameter Description
Name	The name of the API Application. This parameter is required.
Key	The Key used for the API Application. This parameter is required.
Secret	The Secret used for the API Application. This parameter is required.
Enabled	The Enabled parameter tells whether the API Application is enabled or not. This parameter is optional.
CA	The CA parameter sets the CA for the API Application. The format used for this parameter is HostName\\LogicalName. This parameter is optional.
Template	The Template parameter sets the template that is used with the API Application. This should be the template short name. This parameter is optional.

Example Request

POST http://<host>/CMSApi/ApiApp/1/AddApiApp HTTP/1.1

```
{
        "Name": "<Name>",
        "Key": "<hexadecimal key>",
        "Secret": "<hexadecimal secret>",
        "Enabled": true,
        "CA": "<CA Host Name>\\<CA Logical Name>",
```

```
        "Template": "<Template Common Name>"
}
```

Example Response

Status Code: 200

```
{
        "Id": <Id>
}
```

2.3.2.3 ApiApp EditApiApp

The POST EditApiApp endpoint allows certain aspects of an API Application definition to be updated. The only aspect of the API Application that cannot be updated is the Id. The response has the same elements as the GetApiApps call except for a single Api Application. Table 9 – EditApiApp Request Parameters holds the Parameters for the request.

Table 610: AddApiApp Parameters

Parameter Name	Parameter Description
Id	The Id of the API Application that is to be updated. This parameter is required.
Name	The name that the API Application will be updated to. This parameter is optional.
Key	The Key that the API Application will be updated to. This parameter is optional.
Secret	The Secret that the API Application will be updated to. This parameter is optional.
Enabled	The Enabled state the API Application will be updated to. This parameter is optional.
CAId	The Id of the Certification Authority the API Application will be updated to. This is an alternative to CaConfiguration. This parameter is optional.
CaConfiguration	The CA Configuration the API Application will be updated to. The format for the Configuration is Host.Name\\Logical-Name. This is an alternative to CAId. This parameter is optional.
TemplateId	The Id of the Template that the API Application will be updated to. This is an alternative to TemplateName. This parameter is optional.
TemplateName	The Name of the Template the API Application will be updated to. The name of the template should be the short name. This is an alternative to Template Id. This parameter is optional.

Example Request

POST http://<host>/CMSApi/ApiApp/1/EditApiApp

```
{
       "Id": <Id>,
       "Name": "<Name>",
       "Key": "<hexadecimal key>",
       "Secret": "<hexadecimal secret>",
       "Enabled": true,
       "CAId": <CA Id>,
       "TemplateName": "<Template Common Name>"
}
```

Example Response

Status Code: 200

```
{
       "Id": "<Id>",
       "Name": "<Name>",
       "Key": "<hexadecimal key>",
       "Secret": “<hexadecimal secret>”,
       "Enabled": "True",
       "CAId": "<CA Id>",
       "CAConfiguration": "<CA Host Name>\\<CA Logical Name",
       "TemplateId": "<Template Id>",
       "TemplateName": "<Template Common Name>",
       "TemplateForest": " <Template Forest>"
}
```

2.3.2.4 ApiApp DeleteApiApp

The POST DeleteApiApp endpoint removes an API Application from Keyfactor Command. The POST request must contain a JSON string containing the Identity Id. This method returns a 200 a message stating the API App was deleted successfully.

Example Request

POST http://<host>/CMSApi/ApiApp/1/DeleteApiApp HTTP/1.1

```
{
       "Id": <Id>
}
```

Example Response

Status Code: 200

```
{
       "Message": "The Api Application was deleted"
}
```

2.3.3 CertEnroll

The CertEnroll component of the Keyfactor Web APIs includes all methods necessary to programmatically request and obtain a certificate. Keyfactor Command supports enrollment through Microsoft Active Directory Certificate Services Certificate Authorities, both in the local Active Directory forest and, by using Keyfactor Gateways, in remote domains and a variety of public CA vendors. Contact your Keyfactor representative for more information about Keyfactor Gateways, including the most recent list of supported Certificate Authorities.) The CertEnroll component allows enrollment through all CAs configured in your Keyfactor Command environment. The API supports two variations of enrollment. The more secure variant allows the client application to generate the certificate's public/private keypair on the device issuing the request, so that the private key is never transmitted or stored anywhere else. This model is useful in scenarios where the key doesn't need to be archived or exported. The second model lets the server generate the keys, returning the resulting cert and keypair as a PFX/PKCS12 blob. This method is suitable when the key does need to be exported or archived, or when the client is not capable of generating a keypair itself.

There are three versions of the CertEnroll API, each with separate methods for the two enrollment variations, and up to three auxiliary methods to help formulate a successful enrollment request or perform related operations. The complete set of endpoints is given here in Table 611: CertEnroll Endpoints.

Table 611: CertEnroll Endpoints

Endpoint	Method	Description
/1/Status	GET	A synonym for GET /Status, included on this path for backwards-compatibility
/1/Templates	GET	Return a list of certificate templates available to this API application
/1/Token	GET	Retrieve a temporary authentication token to be used with an enrollment request
/1/Pkcs10	POST	Obtain a certificate by providing a CSR, using a key generated by the client
/1/Pkcs12	POST	Obtain a certificate and private key from Keyfactor Command by providing certain certificate attributes
/2/Status	GET	A synonym for GET /Status, included on this path for backwards-compatibility
/2/Templates	GET	Return a list of certificate templates available to this API application

Endpoint	Method	Description
/2/Token	GET	Retrieve a temporary authentication token to be used with an enrollment request
/2/Pkcs10	POST	Obtain a certificate by providing a CSR, using a key generated by the client
/2/Pkcs12	POST	Obtain a certificate and private key from Keyfactor Command by providing certain certificate attributes
/3/Templates	GET	Return a list of certificate templates available to this API application
/3/Renew	POST	Obtain a new certificate based on content from an existing certificate in Keyfactor Command
/3/Pkcs10	POST	Obtain a certificate by providing a CSR, using a key generated by the client
/3/Pkcs12	POST	Obtain a certificate and private key from Keyfactor Command by providing certain certificate attributes

For historic reasons, slight differences in the template format necessitated differentiating the methods into a "version 1" and "version 2", with the same set of methods. Then, to allow simplification of the built-in security mechanisms, version 3 of these methods was introduced. In most cases, applications should use the CertEnrollv3 methods if taking advantage of this security mechanism (as described below) and CertEnrollv2 if not.

This Keyfactor Command API component supports an optional application authentication feature to restrict the API to selected third-party software clients. It uses a public application key and a private application secret. The application key identifies the API client application to the server and is sent as part of the HTTP headers for all enrollment endpoints. The application secret is used to compute an HMAC-SHA1 signature that is sent in an HTTP header for certain endpoints. The combination of the application key and the computed signature allows Keyfactor Command to verify the origin and the authenticity of the enrollment request. Although Basic authentication credentials are required in order to connect to the API, this allows a single user to configure different applications for different templates and have the restrictions enforced. The secret allows secure authentication and prevents attackers from attempting to replay successful enrollment requests. The calculation of this HMAC signature differs between v2 and v3 of the API. The different computations are covered in Table 612: CertEnroll Security Headers.

Another difference between v1, v2 and v3 is that v3 will import the certificate immediately and sync the row from the CA database after the certificate has been issued, whereas v1 and v2 require a manual import of the certificate after it has been issued.

Each application should have its own unique application key and secret pair embedded in the application, as well as in secure storage on the server. These keys can be registered in the API Applications section of the System Settings menu on the Keyfactor Command Management Portal. Giving each application its own key and secret pair provides these advantages:

- An application can be restricted to request specific certificate templates and from specific CAs.
- One application key can be disabled while leaving other application keys enabled. This allows insecure or compromised versions of an application to be disabled without affecting up-to-date users.

Table 612: CertEnroll Security Headers

Header Name	Header Value
X-CSS-CMS-AppKey	This header contains the application key assigned to this particular application. This header is a base-64-encoded string created from the key's byte sequence, and not the ASCII/UTF-8 hexadecimal representation of that byte sequence. For example, if the key is entered in the API Applications section as "030303030303030303FF", this represents the bit pattern "00000011000000110000001100000011000000110000001100000011000000111111111", with base-64 encoding "AwMDAwMDAwMD/w==". In Python, this conversion can be accomplished with the following code: import base64 hexKey = "030303030303030303FF" binKey = hexKey.decode("hex")) b64Key = base64.b64encode(binKey)
X-CSS-CMS-Token	This header field contains the temporary token that was previously obtained from the GET Token method. Like the application key, this header is a base-64-encoded string created from the binary form of the token, and not the ASCII/UTF-8 hexadecimal representation actually returned by the response to the GET Token. This is required for the v1 and v2 endpoints only.
X-CSS-CMS-Signature	This header field contains an HMAC-SHA-1 message signature computed from the request. Producing this signature proves that the client has access to the application secret value that is also present in the server's configuration, that the message has been transmitted without modification, and that transmission is recent. This is required for all enrollment endpoints, although this requirement can be disabled through the application settings in the management console. The computation of this signature differs between versions; all versions are a base-64 encoding of a SHA-1 hash, but the content to be hashed varies. In general, v1 and v2 GET methods hash the URL and Token; v1 and v2 POST methods hash the URL, Token, and request body; v3 GET methods do not require a signature; and v3 POST methods hash the request body only. The computation of HMAC signatures is significantly easier for v3 methods. Sample Python code is included in Table 613: CertEnroll HMAC computations in Python for each computation type.

Table 613: CertEnroll HMAC computations in Python

Endpoints	Signature
GET Templates (v1 and v2)	token = json.loads(GET_Token_ResponseBody)["SessionTokenValue"] URLPath = "/CMSApi/CertEnroll/2/Templates" requestDataString = URLPath + token; appSecretBytes = appSecretString.decode("hex") signature = hmac.new(appSecretBytes, requestDataString, hashlib.sha1).hexdigest(); headers["X-CSS-CMS-Signature"] = base64.b64encode(signature.decode("hex"));
POST	token = json.loads(GET_Token_ResponseBody)["SessionTokenValue"]

Endpoints	Signature
/1/Pkcs10, /1/Pkcs12, /2/Pkcs10, /2/Pkcs12	URLPath = "/CMSApi/CertEnroll/2/Pkcs12" body='{"Flags":0,"TemplateName":"User","Pkcs12Password":"lily1234","SubjectNameAttributes":null}' **Note:** For these methods, the body must be formatted exactly as above, as far as parameter order, capitalization, and whitespace. This is one reason v3 signatures are easier to use. requestDataString = URLPath + token + body appSecretBytes = appSecretString.decode("hex") signature = hmac.new(appSecretBytes, requestDataString, hashlib.sha1).hexdigest(); headers["X-CSS-CMS-Signature"] = base64.b64encode(signature.decode("hex"));
POST /3/Pkcs10 and /3/Pkcs12	data='{"Flags":0,"TemplateName":"User","Pkcs12Password":"lily1234","SubjectNameAttributes":null}' body = '{"Timestamp" : '" + datetime.datetime.utcnow().isoformat()+ '", "Request" : ' + data + '}' **Note:** For these methods, the request can be formatted in any equivalent json format without regard to capitalization, whitespace, or order of elements. This is one reason v3 signatures are easier to use. appSecretBytes = appSecretString.decode("hex") signature = hmac.new(appSecretBytes, body, hashlib.sha1).hexdigest(); headers["X-CSS-CMS-Signature"] = base64.b64encode(signature.decode("hex"));

2.3.3.1 CertEnroll Token

The GET Token request returns a session token that is used in subsequent calls to v1 or v2 enrollment endpoints to authenticate the software client to the server. By default, the token has an expiration time of 10 minutes (configurable in the Keyfactor Command Management Portal Application Settings). Using a token after it is expired will result in an error.

Example Request

GET http://<host>/CMSApi/CertEnroll/1/Token HTTP/1.1

Example Response

```
{
        "SessionTokenValue":"F715F307DBE0DD5A9894260DBF0643C042173698"
}
```

2.3.3.2 CertEnroll Templates

The Templates methods return the list of templates configured and enabled for use by the application (identified by the X-CSS-CMS-AppKey HTTP header). The set of fields returned for a template differs from version 1 to version 2, but versions 2 and 3 return the same content. No parameters are required for these requests—only the app key in the header, formatted as described in Table 612: CertEnroll Security Headers—but the response formats are given in Table 614: GET /2/Templates and /3/Templates Response Body.

Important: As of release 9.0 of the Classic API, version 1 of CertEnroll/1/Templates has been removed from the product and is no longer supported.

Table 614: GET /2/Templates and /3/Templates Response Body

Parameter Name	Parameter Value
DisplayName	Long/Friendly name of the template.
CommonName	Short name of the template.
Oid	Object Identifier for this template.
KeySize	String representation of Key Size in bits, or Unknown.

Example Request

GET http://<host>/CMSApi/CertEnroll/3/Templates HTTP/1.1

X-CSS-CMS-AppKey: AAAAAAAAAAAAAA==

X-CSS-CMS-Token: A0sTeMd9PT6XPw2BdqWb9PkErQk= [Version 2 only]

Example Response

Version 2 and 3

```
[
	{
		"DisplayName":"UserServer",
		"CommonName":"UserServer",
		"Oid":"1.3.6.1.4.1.311.21.8.2290866.14924250.4277929.6978074.6651290.247.14988018.16169587",
		"KeySize":"2048"
	}
]
```

2.3.3.3 CertEnroll Pkcs10

The PKCS10 method provides enrollment with on-device key generation. The basic workflow with on-device key generation is:

1. Client application retrieves list of available certificate templates using the Keyfactor Command API.
2. Client generates a public/private key pair based on the key size requirements from the selected template.
3. Client creates a PKCS10 Certificate Signing Request (CSR) using the keypair and template attributes.
4. Client sends the PKCS10 request and selected template name to the API which submits the request to the enterprise CA and returns the certificates received from the CA to the software client.

If successful, the response from the CA will be a PKCS#7 message containing the issued certificate and (optionally) the certificate chain. Once the response is received, a software client can construct a PKCS12 package with the previously generated key pair and the issued certificates, import the keys and certificates into an application-specific store, such as Apple's KeyChain Services or a Java Keystore, or perform any other processing required. The flow (for versions 1 and 2) is shown in Figure 6: Pkcs#10-Based Enrollment Request. The version 3 flow is identical except that a token is not required for enrollment, so the initial exchange with the *token* endpoint is not needed. The difference in version 3 is explained in Table 612: CertEnroll Security Headers.

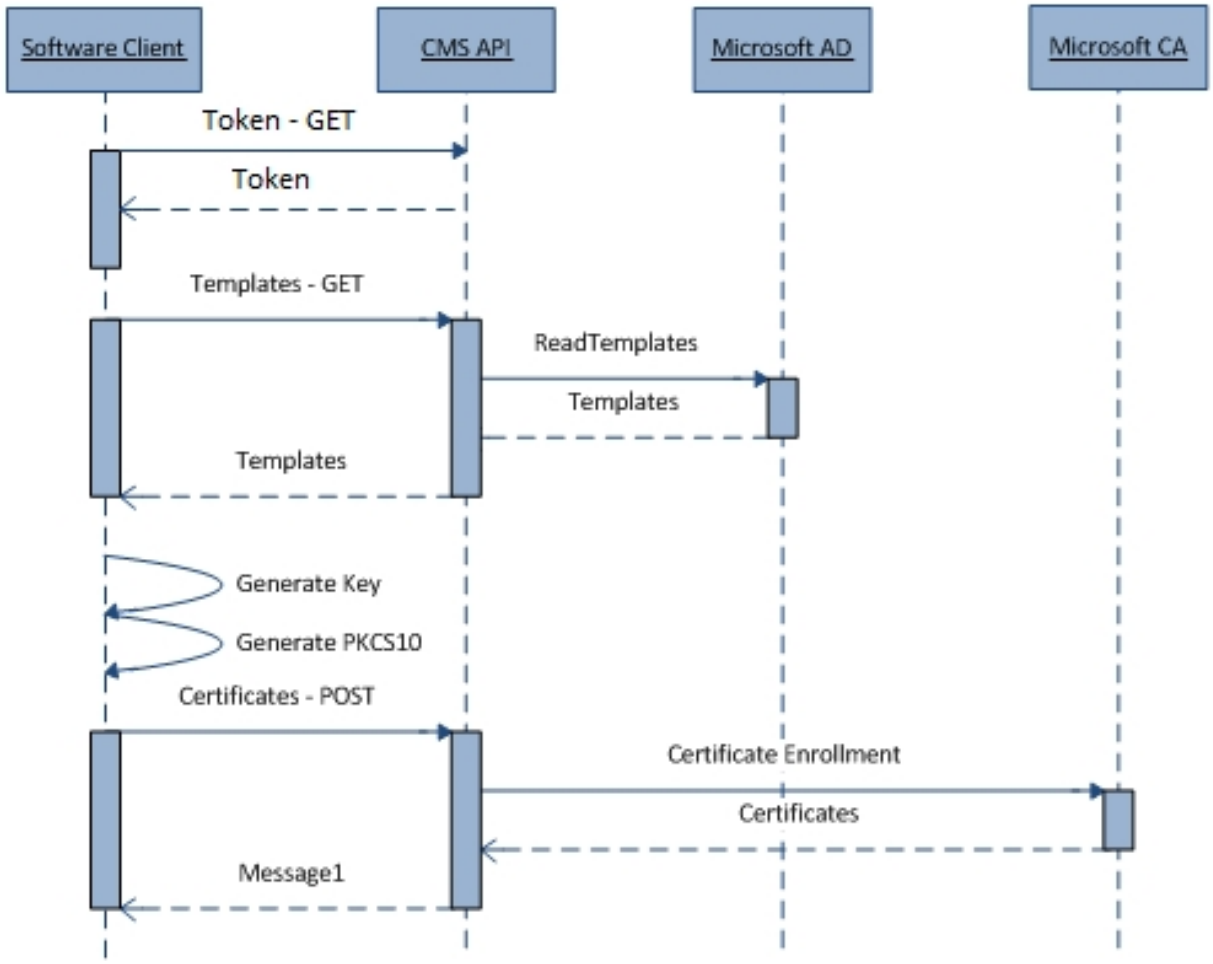


Figure 6: Pkcs#10-Based Enrollment Request

The PKCS10-based method is the most secure way to enroll for certificates with Keyfactor Command. The PKCS10 method utilizes on-device key generation instead of the server-based key generation used for the PKCS12 method. The PKCS10 method also requires the use of a certificate template that populates the subject and/or subject alternate name from Active Directory. This reliance on certificate templates allows Keyfactor Command to utilize the security mechanisms built into the Microsoft CA Services.

To use this method, the following configuration should be present on the Keyfactor Command server and in its domain:

- IIS application pool configured with a non-administrator domain member account
- API Application with valid key, secret, template, and CA
- Certificate template configured to:
    - Populate subject and/or subject alternate name (SAN) fields from AD as needed. While the PKCS#10 request may contain data for these fields, the selected certificate template may replace those values with information from Active Directory.

- Not allow private key exportation
- Grant enroll permission to all users who may enroll for a certificate

The request parameters that should be sent for version 1 and 2 of the enrollment are listed in Table 615: POST /1/Pkcs10 and /2/Pkcs10 Request Body and for version 3 in Table 616: POST /3/Pkcs10 Request Body, while the response format (for all versions) is given in Table 617: POST /*/Pkcs10 Response Body:

Table 615: POST /1/Pkcs10 and /2/Pkcs10 Request Body

Parameter Name	Parameter Value
Flags	Bit flags that determine the enrollment behavior. At this time, the only available bit flag is: • 0x01 = Include certificate only (default is to return certificate + trust chain)
TemplateName	Name of the certificate template to use for enrollment. This name must match one of the template names configured for this application key in the API Applications page.
Pkcs10Request	Contains a base-64-encoded PKCS#10 request generated on the device. The key sizes used to generate the PKCS#10 request must match the key size specified in the certificate template.
MetadataList	A list of key value pairs for each metadata item that is to be set on the issued certificate in Keyfactor Command. This parameter is optional.

In version 3 of the API, the fields used by versions 1 and 2 are wrapped in an outer envelope and sent along with a timestamp. By including this timestamp in the request body and using this as part of the HMAC signature computation, the need for a current API access token is eliminated without reducing security. The request structure for version 3 of this endpoint is shown in Table 616: POST /3/Pkcs10 Request Body:

Table 616: POST /3/Pkcs10 Request Body

Parameter Name	Parameter Value
Timestamp	ISO 8601 Timestamp in UTC timezone, e.g. "2018-11-22T20:41:08.440Z"
Request	JSON object in the same format as a version 1/2 Pkcs10 enrollment request (see Table 615: POST /1/Pkcs10 and /2/Pkcs10 Request Body).

Table 617: POST //Pkcs10 Response Body*

Parameter Name	Parameter Value
SerialNumber	String containing the hexadecimal serial number of the issued certificate.
IssuerDN	Distinguished Name of the certificate's issuer.

Parameter Name	Parameter Value
Thumbprint	Thumbprint of the issued certificate.
CMSID	Identifier for this certificate in Keyfactor Command. Can be used to identify the cert in future API requests.
CMSRequestId	Identifier for the certificate request in Keyfactor Command. Can be used if certificate is pending issuance.
Certificates	If the CERT_ONLY flag (0x01) is set in the request, then the response is a base-64 encoding of the DER-encoded cert. If the CERT_ONLY flag is not set, the response is a base-64 encoding of a PKCS7 containing the cert and its chain.
RequestDisposition	Value returned by the CA in response to this certificate request
DispositionMessage	Message accompanying the disposition value returned by the CA

Example Request

Versions 1 and 2

POST http://<host>/CMSApi/CertEnroll/1/Pkcs10 HTTP/1.1

```
{
	"Flags":0,
	"TemplateName": "User",
	"Pkcs10Request":"-----BEGIN CERTIFICATE REQUEST-----
		<base64-encoded-certificate-request>
		-----END CERTIFICATE REQUEST-----\n"
}
```

Example Request

Version 3

```
{
	"Timestamp" : "2017-12-18T19:56:12.365Z",
	"Request": {
		"Flags":0,
		"TemplateName": "User",
		"Pkcs10Request":"-----BEGIN CERTIFICATE REQUEST-----
		<base64-encoded-certificate-request>
		-----END CERTIFICATE REQUEST-----",
```

```
               "MetadataList":{"<metadata type name>":"<metadata value>","<metadata type name>":"<metadata
value">}
       }
}
```

Example Response

```
{
       "SerialNumber": "2684C97728678A944A67C03E7192785B",
       "IssuerDN": "CN=CorpCA1, DC=keyexample, DC=com",
       "Thumbprint": "FDB3A0F4ADCF9C39A2BB639898EE1670DFDBF5BB",
       "CMSID": 5,
       "CMSRequestId": 3,
       "Certificates": <PEM-encoded certificates>
       "RequestDisposition": "Issued",
       "DispositionMessage": ""
}
```

2.3.3.4 CertEnroll Pkcs12

The PCKS12-based POST enrolls for a certificate with a server-generated private key. It generates a PKCS#12 file that is protected by the password specified in the request and returns a base-64-encoded PKCS#12 response if successful.

The basic workflow with server-based key generation is:

1. Third-party software client retrieves a list of available certificate templates using the Keyfactor Command API.
2. Third-party software client sends the selected template name and a password to the API. The Keyfactor Command component will:
    a. Generate the RSA key pair.
    b. Submit the request to the CA configured for the API application and retrieve the issued certificate.
    c. Create a PKCS12 blob with the private key, the issued certificate, and the certificate trust chain using the supplied password.
    d. Return the PKCS12 blob to the API client.

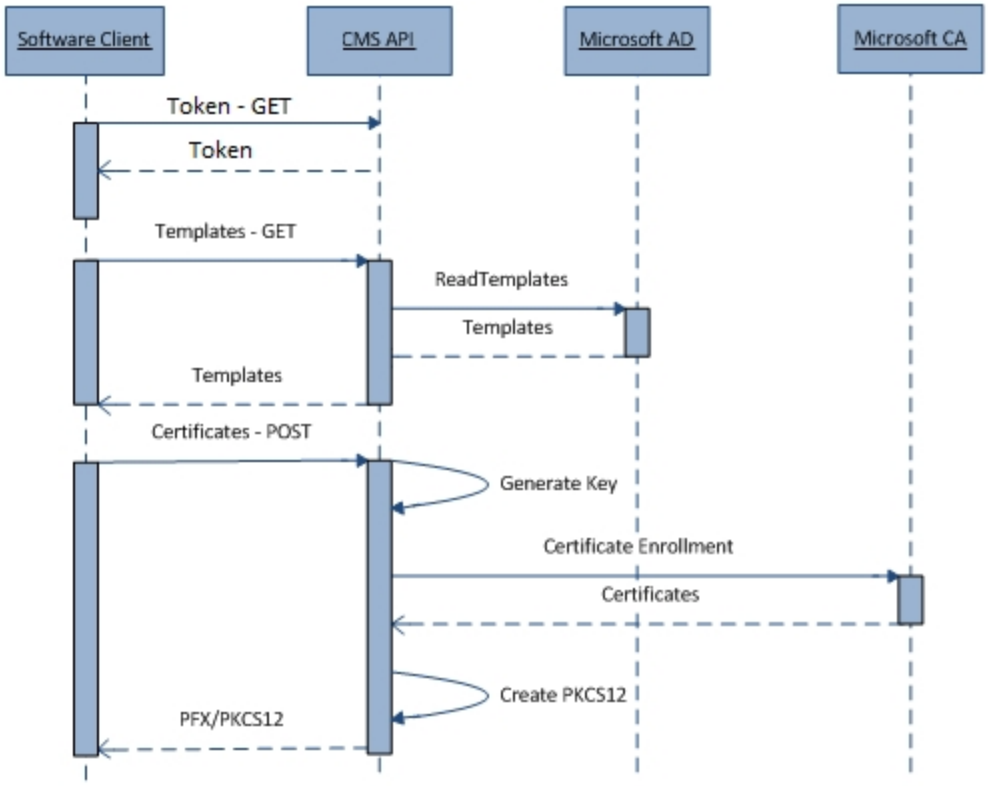


Figure 7: Pkcs#12-Based Enrollment Request

As with the Pkcs10 methods, versions 1 and 2 use the flow shown in Figure 7: Pkcs#12-Based Enrollment Request, while version 3 does not require a token and uses a timestamp instead. For these methods, keys are generated on the server and returned to the client in the form of a P12 (PFX) file. This requires that the certificate's private key is transmitted over the network and temporarily stored on the server, which can present a security risk. For this reason, Keyfactor recommends that clients which are capable of generating their own keypair and submitting a CSR use the Pkcs10 enrollment. When clients do not have the capability or the processing power to do this, the Pkcs12 offers an alternate method. Certificate templates are used on the Microsoft CA; however, the private key must be marked as exportable in the template.

To use this method, the following configuration needs to be present on the Keyfactor Command server:

- Certificate template configured to:
  - Allow the requestor (Keyfactor Command) to supply the subject and subject alternate name details
  - Allow the private key to be exported
  - Grant enroll permission to the Keyfactor Command application pool user—no other user needs enroll permissions for this template and for best security, none should be granted to other users
- IIS application pool user configured to be a non-administrative domain member account
- The *Load User Profile* option configured to **true** under the advanced settings for the application pool

The format of a Pkcs12 request is given in Table 618: POST /1/Pkcs12 and /2/Pkcs12 Request Body and Table 619: POST /3/Pkcs12 Request Body, while the response format is given in Table 620: POST /*/Pkcs12 Response Body.

Table 618: POST /1/Pkcs12 and /2/Pkcs12 Request Body

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td>Flags</td><td>Bit flags that determine the enrollment behavior. At this time, there are no available bit flags so this value should be set to "0" (zero).</td></tr>
<tr><td>TemplateName</td><td>Name of the certificate template used for enrollment. This name must match- one of the allowed template names.</td></tr>
<tr><td>Pkcs12Password</td><td>PKCS12 password. Must be 8 or more characters.</td></tr>
<tr><td>SubjectNameAttributes</td><td>Token values that are substituted into the API subject format string. When needed values are not provided, Keyfactor Command will attempt to use the corresponding field from the requester’s AD account. If no attributes are needed in the request, you must still include this attribute and set the value to null. Also, note that, although the terms are similar, SubjectNameAttributes are NOT the same as Subject Alternative Names, which are supplied separately in the SubjectAltNameElements field.
Values can be supplied either as an array of key/value pairs or as a dictionary in the form {"Field1" : "Value1", "Field2" : "Value2"}.</td></tr>
<tr><td>SubjectAltNameElements</td><td>Contains an array of key/value pairs that represent the elements for Keyfactor Command to use when generating the certificate’s subject alternative name. This parameter is optional. The key will be the numeric subject alternative name flag (in string form), associated with the value. The valid subject alternative name flags are as follows:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>1</td><td>UPN</td></tr>
<tr><td>2</td><td>RFC822</td></tr>
<tr><td>3</td><td>DNS</td></tr>
<tr><td>4</td><td>IP Address</td></tr>
<tr><td>5</td><td>URI</td></tr>
<tr><td>6</td><td>Email</td></tr>
<tr><td>7</td><td>GUID</td></tr>
<tr><td>8</td><td>Registered Id</td></tr>
<tr><td>9</td><td>Directory name</td></tr>
</table></td></tr>
<tr><td>MetadataList</td><td>A list of key value pairs for each metadata item that is to be set on the issued certificate in Keyfactor Command. This parameter is optional.</td></tr>
</table>

Table 619: POST /3/Pkcs12 Request Body

Parameter Name	Parameter Value
Timestamp	ISO 8601 Timestamp, e.g. "2018-11-22T20:41:08.440000"
Request	JSON object in the same format as a version 1/2 Pkcs10 enrollment request (see Table 618: POST /1/Pkcs12 and /2/Pkcs12 Request Body).

Table 620: POST //Pkcs12 Response Body*

Parameter Name	Parameter Value
SerialNumber	String containing the hexadecimal serial number of the issued certificate.
IssuerDN	Distinguished Name of the certificate's issuer.
Thumbprint	Thumbprint of the issued certificate.
CMSID	Identifier for this certificate in Keyfactor Command. Can be used to identify the cert in future API requests.
CMSRequestId	Identifier for the certificate request in Keyfactor Command. Can be used if certificate is pending issuance.
Pkcs12Blob	Base-64-encoded representation of the Pkcs#12 certificate that was issued, if any.
RequestDisposition	Value returned by the CA in response to this certificate request.
DispositionMessage	Message accompanying the disposition value returned by the CA.

This method allows additional attributes to be included in the certificate's subject name. A common use for this is the inclusion of a device class identifier, such as "iPhone 4S". On the Keyfactor Command server there is a configuration property to define the format of the subject name. An example is:

```
"CN={cn},OU=Device Model {deviceType}"
```

For each of the tokens given in {brackets}, Keyfactor Command will replace the value with the corresponding value in the SubjectNameAttributes field of the request, if present. If no value is provided, it will attempt to look up the value in the requester's AD account. In this example, Keyfactor Command might replace the string "{deviceType}" with attribute value "deviceType" supplied in the SubjectNameAttributes key-value-pair structure inside of the JSON request from the API client, and (if "cn" is not specified in the request) the "{cn}" string would be replaced with the value of the "cn" property from the user's Active Directory properties. If a matching token cannot be found either in the request or in AD, no value is substituted.

Example Request

Versions 1 and 2

POST http://<host>/CMSApi/CertEnroll/2/Pkcs12 HTTP/1.1

```
{
       "Flags":0,
       "Pkcs12Password": "12341234",
       "TemplateName": "User",
       "SubjectNameAttributes": {"deviceid":"iPad"}]
}
```

Example Request

Version 3

POST http://<host>/CMSApi/CertEnroll/3/Pkcs12 HTTP/1.1

```
{
       "Timestamp" : "2017-12-18T19:56:12.365Z",
       "Request": {
              "Flags":0,
              "Pkcs12Password": "12341234",
              "TemplateName": "User",
              "SubjectNameAttributes": [{"key":"deviceid","value":"iPad"}],
              "MetadataList":{"<metadata type name>":"<value>","<metadata type name>":<value>"}
       }
}
```

Example Response

Status Code: 200

```
{
       "SerialNumber": "690003CC096AC71023934747AA00000003CC09",
       "IssuerDN": "CN=jdk-CA1, DC=jdk, DC=com",
       "Thumbprint": "04259811B3BC522093532FBA5F4C1FA3C0969A87",
       "CMSID": 8,
       "CMSRequestId": 6,
       "Pkcs12Blob": <base64-encoded PKCS#12>,
       "RequestDisposition": "Issued",
```

```
      "DispositionMessage": ""
}
```

2.3.3.5 CertEnroll Renew

Certificate renewal in Keyfactor Command allows a certificate to be issued based on data from an existing certificate. Some configurations, such as the issuing CA and template, can be made to differ between the original certificate and the renewed one. At renewal time, the new certificate can also be automatically delivered to different certificate stores managed by Keyfactor Command Agents, replacing the old certificates. This provides an easy mechanism to quickly replace expiring or compromised certificates, migrate deployed certificates from one PKI to another, or replace certificates with similar certificates using more secure cryptographic algorithms. The Renew Web API method, along with the web console and expiration alert handlers, allows access to this renewal functionality. The structure of a renew request is given Table 621: POST /3/Renew Request Body, and the response in Table 622: POST /3/Renew Response Body.

Table 621: POST /3/Renew Request Body

Parameter Name	Parameter Value
Lookup	Description of the certificate to be renewed. See Table 4: Classic API Certificate Lookup Structure.
CertStores	Array of GUIDs listing the certificate stores where the new certificate should be delivered. This must be a subset of the CertStores containing the original certificate.
Template	Certificate template to be used for the new certificate request.
CAConfiguration	Certificate authority for the new certificate, in the form "hostname\\logical name" (double-backslash required for JSON formatting).
Metadata	Optional dictionary of metadata fields and values to be associated with the newly issued certificate.
CustomPassword	Password to protect the private key of the new certificate. This field is optional and Keyfactor Command will use a randomly assigned password if this is not set.

Table 622: POST /3/Renew Response Body

Parameter Name	Parameter Value
Thumbprint	Thumbprint of the issued certificate.
CMSRequestId	Identifier for the certificate request in Keyfactor Command, if certificate is pending issuance.
RequestDisposition	Value returned by the CA in response to this certificate request.

Parameter Name	Parameter Value
DispositionMessage	Message accompanying the disposition value returned by the CA.
RenewedCertStores	List of certstores that had a certificate addition job scheduled successfully. The certstores will be listed in the format "<Store machine >-<Store path>".

Example Request

POST http://<host>/CMSApi/CertEnroll/3/Renew HTTP/1.1

```
{
       "Lookup": {"Type" : "CMSID", "CMSID" : 7},
       "CertStores": ["&lt;Guid&gt;"],
       "Template": "UserServer",
       "CAConfiguration" : "CA1.jdk.com\\jdk-CA1",
       "Metadata":{"Email-Contact":"a.b@example.com"}
}
```

Example Response

```
{
"RenewedCertStores": ["192.168.41.171-/home/pi/cherry/cherrystore"],
"Thumbprint": "46CCE7023bce5c434f4206b74473fd614df56218",
"CMSRequestId": 0,
"RequestDisposition": "Issued",
"DispositionMessage": "The certificate renewal has been completed successfully. Agent jobs to install
the new certificate have been created."
}
```

2.3.4 Certificates

The Certificates component of the Web API supports certificate lifecycle and management tasks apart from enrollment. The complete set of methods in this component is given in Table 623: Certificates Endpoints.

Table 623: Certificates Endpoints

Endpoint	Method	Description
/3/Contents	POST	Return the certificate contents in PEM format
/3/Count	POST	Return the number of certificates in the Keyfactor Command database matching a given search query

Endpoint	Method	Description
/1/Metafield	POST	Associate a metadata value with a certificate in the Keyfactor Command database.
/2/Import	POST	Add an existing certificate into the Keyfactor Command database.
/3/Revoke	POST	Revoke a given certificate.
/3/Recover	POST	Recover a given certificate
/3/PublishCRL	POST	Request a CA to publish a new CRL
/3/Search	POST	Return the full set of certificates in the Keyfactor Command database matching a given search query

2.3.4.1 Certificates Metafield

The metafield POST method is used to import individual certificate metadata field values into Keyfactor Command. This method offers limited functionality and security measures compared to the Metadata v2 and v3 methods described in the Metadata section (see Metadata on page 1443), but is included for backward-compatibility. A JSON string must be submitted with the POST request containing the data shown in Table 624: POST /1/Metafield Request Body. This method returns HTTP 200 OK with message body "true" on success or an appropriate 4xx status with an accompanying error message in the body on failure.

Table 624: POST /1/Metafield Request Body

Parameter Name	Parameter Value
CertificateId	The Keyfactor Command database row identifier associated with the existing certificate. Many times this will be the certificate id returned by the import API call.
MetadataFieldTypeName	The string name of the metadata field type for which the value is provided.
Value	The metadata field value to be associated with the provided certificate identifier.

Example Request

POST http://<host>/CMSApi/Certificates/1/Metafield HTTP/1.1

```
{
       "CertificateId": 1,
       "MetadataFieldTypeName": "Email-Contact",
```

```
      "Value": "support@example.com"
}
```

2.3.4.2 Certificates Import

The certificate import POST method is used to import a certificate (.cer) file into Keyfactor Command while also allowing the simultaneous definition of metadata values for the imported certificate. The POST request must contain a JSON string containing the certificate and any metadata items that should be associated with the certificate but does not require the content-disposition or multi-part form found in version 1. This method returns HTTP 200 OK with message body "true" on success or an appropriate 4xx status with an accompanying error message in the body on failure.

Important: Support for version 1 of the Classic API certificate import method (Certificates/1/Import) will end in an upcoming release of the product. All applications should be migrated to a newer Import endpoint.

Table 625: POST /2/Import Request Body

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td>X509Base64</td><td>String containing the certificate blob. This may include the "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" references or these may be left out.</td></tr>
<tr><td>MetadataList</td><td>A comma-delimited list of metadata fields, each containing two parts:
<table><tr><th>Name</th><th>Description</th></tr><tr><td>MetadataFieldTypeName</td><td>The metadata field name—e.g. Email-Address</td></tr><tr><td>Value</td><td>The metadata value—e.g. bob.smith@example.com</td></tr></table>The metadataList parameter is not required, but if you choose to include it, you must include both the name and the value for each metadata value to be imported.</td></tr>
<tr><td>CertState</td><td>Used to manually set the state of the imported certificate. The following values are accepted:
<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></table></td></tr>
</table>

Parameter Name	Parameter Value
	<table><tr><th>Value</th><th>Description</th></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>

Example Request

POST http://<host>/CMSApi/Certificates/2/Import HTTP/1.1

```
{
       "x509Base64": "-----BEGIN CERTIFICATE-----
       <base64-encoded-certificate-contents>
       -----END CERTIFICATE-----",
       "MetadataList":[
              {"MetadataFieldTypeName": "Email-Contact",
              "Value":"john.doe@example.com"}
       ]
}
```

2.3.4.3 Certificates Contents

The Contents method retrieves the contents of a specified certificate. The request requires only enough information to identify a certificate in Keyfactor Command, and the body of a successful response will consist solely of the PEM-encoded representation of that certificate. Unlike most methods, for successful requests the response content type will be "text/plain".

Table 626: POST /3/Contents Request Body

Parameter Name	Parameter Value
Lookup	Description of the certificate to be retrieved. See Table 4: Classic API Certificate Lookup Structure.

Example Request

POST http://<host>/CMSApi/Certificates/3/Contents HTTP/1.1

```
{
       "Lookup": {"type": "CMSID", "CMSID": <cms-certificate-id>}
}
```

Example Response

```
<base64-encoded-certificate-contents>
```

2.3.4.4 Certificates PublishCRL

The PublishCRL method will cause Keyfactor Command to make a request to the provided Certificate Authority to publish a new CRL to the locations configured by the CA. This method requires only a single parameter and returns no response body on a successful request. On an unsuccessful request, an appropriate HTTP status code along with a string in the response body describing the error is returned.

Table 627: POST /3/PublishCRL Request Body

Parameter Name	Parameter Value
CertificateAuthority	Certificate authority for the new CRL, in the form "hostname\\logical name" (double-backslash required for JSON formatting).

Example Request

POST http://<host>/CMSApi/Certificates/3/PublishCRL HTTP/1.1

```
{
       "CertificateAuthority" : "CA1.corp.com\\Issuing-CA1"
}
```

2.3.4.5 Certificates Recover

The Recover method allows a user to recover an archived private key for an issued certificate. For recovery to succeed, the CA that issued the certificate must have been configured to archive the private key, and the Key Recovery Agent certificate must be imported into the personal certificate store of the Keyfactor Command API IIS Application Pool's user account on the Keyfactor Command API server. If successful, the method will return the certificate and recovered private key as a base64-encoded PFX file. On error, an appropriate HTTP status code and message will be returned. See *Configuring Key Recovery for Keyfactor Command* in the *Keyfactor Command Reference Guide* for information about configuring key recovery.

Table 628: POST /3/Recover Request Body

Parameter Name	Parameter Value
Lookup	Description of the certificate to be renewed. See Table 4: Classic API Certificate Lookup Structure.
Details	Information to complete the recovery operation. This contains just a single field: **Parameter Name** / **Parameter Value** Password / Password for the archived private key.

Example Request

POST http://<host>/CMSApi/Certificates/3/Recover HTTP/1.1

```
{
       "Lookup" : {"Type" : "CMSID", "CMSID" : 248852},
       "Details": {"Password": "MyPassword1234"}
}
```

Example Response

```
{
       "pfx" : "<PEM-encoded pfx>"
}
```

2.3.4.6 Certificates Revoke

The Revoke method will attempt to revoke a certificate stored in Keyfactor Command. The certificate to be revoked can be identified using the *lookup* request body parameter (see Table 4: Classic API Certificate Lookup Structure). In addition, the message may contain string parameters describing the revocation. Caution is advised when programmatically revoking certificates as the operation generally cannot be undone. The method returns a 200 OK response if successful or an appropriate HTTP code and error message if unsuccessful.

Table 629: POST /3/Revoke Request Body

Parameter Name	Parameter Value
Lookup	Criteria to specify the certificate to be revoked. See Table 4: Classic API Certificate Lookup Structure.
Details	Details used to define the revocation operation. See Table 630: Certificate Revocation Details.

Table 630: Certificate Revocation Details

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td>Reason</td><td>Integer code for certificate revocation reason, as per IETF RFC 5280 ReasonFlags. This field is optional and will default to "0" (zero - unspecified). Allowed values are listed below:
<table>
<tr><th>Value</th><th>Definition</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>
</table></td></tr>
<tr><td>Comment</td><td>Explanation of revocation reason. Optional and will default to the empty string "".</td></tr>
<tr><td>EffectiveDate</td><td>Date on which the revocation will take effect. Optional and will default to the current time if not specified.</td></tr>
<tr><td>noCRL</td><td>If provided and set to "true", Keyfactor Command will not attempt to have the CA publish a new CRL. Optional and treated as "false" by default.</td></tr>
</table>

Example Request

POST http://<host>/CMSApi/Certificates/3/Revoke HTTP/1.1

```
{
       "Lookup": {"Type": "CMSID", "CMSID": 45},
       "Details": {"Reason":4, "EffectiveDate" : "2017-12-29", "Comment": "Reissued 12-27"}
}
```

2.3.4.7 Certificates Search and Count

The Search method will return the set of certificates known to Keyfactor Command that satisfy certain criteria. The criteria that can be searched on and the syntax by which queries are formed is the same as in the Advanced Certificate Search within the Keyfactor Command Management Portal. This is largely consistent with PowerShell comparison notation, but Keyfactor does not publish a complete specification of this query language. Instead,

developers are encouraged to examine the query strings formed in the Keyfactor Command Management Portal and model their API queries based on this. The response will contain a JSON body with an array whose entries each represent a single matching certificate. The Count method expects the same parameters as the Search query but simply returns a count of the records that would be returned if the same parameters were provided to the Search endpoint. For Count, the sorting parameters will have no effect.

Table 631: POST /3/Search and /3/Count Request Body

Parameter Name	Parameter Value
IncludeRevoked	Boolean denoting if revoked certificates should be included in the search results.
IncludeExpired	Boolean denoting if expired certificates should be included in the search results.
Query	Search query criteria, as defined above.
SortField	Name of the result field by which the results should be sorted. The field must be one returned within the results. This parameter is optional and the Keyfactor Command certificate id will be used if not provided. The available fields are the same as in Table 632: POST /3/Search Response Body.
SortAscending	Boolean value denoting if the SortField should be sorted in ascending order. This parameter is optional and ascending will be used if not provided.
SkipCount	Number of records that should be skipped in the results, starting from the beginning of the records (for pagination). This field is optional and no records will be skipped if not provided.
ReturnLimit	Numeric value of the limit of records to be returned. This field is optional and 5000 will be used if not provided.

Table 632: POST /3/Search Response Body

Parameter Name	Parameter Value
Id	Certificate ID assigned by Keyfactor Command, which can be used for service chaining to other many other Web API requests by providing this value as a *CMSID* in the *Lookup* section of the request. See Table 4: Classic API Certificate Lookup Structure.
IssuedCN	Issued Common Name
IssuedDN	Issued Distinguished Name
NotBefore	Beginning date for certificate validity
NotAfter	Ending (expiration) date for certificate validity
IssuerDN	Issuer Distinguished Name

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td>PrincipalName</td><td>Subject Principal Name</td></tr>
<tr><td>RequesterName</td><td>Requester Name</td></tr>
<tr><td>TemplateName</td><td>Certificate Template Name</td></tr>
<tr><td>CertState</td><td>Certificate State. Will take one of the following values:<table><tr><th>Value</th><th>Definition</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>CertificateAuthority</td></tr><tr><td>7</td><td>ParentCertificateAuthority</td></tr></table></td></tr>
<tr><td>KeySize</td><td>Bit-length of the public/private keys.</td></tr>
<tr><td>KeyType</td><td>Cryptographic algorithm used for the public/private key. Will take one of the following values:<table><tr><th>Value</th><th>Definition</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>SerialNumber</td><td>The hexadecimal serial number of the certificate.</td></tr>
<tr><td>Thumbprint</td><td>The hexadecimal thumbprint of the certificate.</td></tr>
</table>

Example Request

POST http://<host>/CMSApi/Certificates/3/Search HTTP/1.1

```
{
       "includeRevoked": true,
       "includeExpired": true,
       "query": "(ExpirationDate -eq \"2018-05-10\")"
}
```

Example Response

```
[{
       "Id":<certificate-id>,
       "IssuedCN": "<cn>",
       "IssuedDN": "<dn>",
       "NotBefore": "2017-05-10T18:59:57",
       "NotAfter": "2018-05-10T18:59:57",
       "IssuerDN": "<issuer-dn>",
       "PrincipalName": null,
       "RequesterName": null,
       "TemplateName": null,
       "CertState": 0,
       "KeySize": 4096,
       "KeyType": 1
}]
```

2.3.5 Certstore

The Certstore Web API (formerly known as the Jks API) provides a set of methods to support management of certificate locations. Keyfactor Command currently supports management of certificates in the following remote locations:

- Java Keystore
- PEM file
- F5 BigIP Web Server
- F5 BigIP SSL Profiles
- Windows Machine Personal, Revoked, and Trusted Roots stores
- Citrix NetScaler virtual servers

Keyfactor Command can, through different Keyfactor Command Agents and Orchestrators, inventory, install, and remove certificates for each of these store types. For certain store types, additional actions are supported as well. The certstore API provides a way to programmatically schedule jobs for these stores. For more information about certificate stores and their support within Keyfactor Command, see the Reference Guide and Installing Orchestrators guide, or contact your Keyfactor representative. This API component currently has only one version, but for

backward-compatibility, it can be accessed through the component name "Certstore" (e.g. /CMSApi/Certstore/1/AddCert) or the legacy name "Jks" (e.g. /CMSApi/Jks/1/AddCert). The set of methods in this API component that can be used to manage certificate stores and their scheduled jobs is listed below in Table 633: Certstore Endpoints.

Table 633: Certstore Endpoints

Endpoint	Method	Description
AddCert	POST	Add given certificate (without private key) to a given certificate store (as well as Keyfactor Command)
AddCertStore	POST	Define a new certstore in Keyfactor Command
AddCertStoreServer	POST	Define a new remote server (e.g. F5, NetScaler) in Keyfactor Command to be managed by a Keyfactor Command agent
AddPFX	POST	Add a PFX file (with private key) to a given certificate store (as well as Keyfactor Command)
AddCertStoreType	POST	Add a Certificate Store Type to be used by a certificate store
CreateJKS	POST	Create a Java Keystore on the file system on target machine
EditCertStore	POST	Update a definition of an existing certificate store in Keyfactor Command
EditCertStoreServer	POST	Update a definition of an existing remote server managed by a Keyfactor Command agent
GetCertStoreTypes	GET	List all certificate store types
Inventory	POST	Retrieve the inventory of a given certificate store
Keystores	GET	Get a list of certificate stores defined in Keyfactor Command
Remove	POST	Remove a certificate from a certificate store
ScheduleInventory	POST	Schedule a certificate store inventory job schedule
ScheduleJob	POST	Schedule a certificate store management job
Status	GET	A synonym for GET /Status, included on this path for backwards-compatibility

2.3.5.1 CertStore AddCert

The POST AddCert method will schedule the addition of the provided certificate to the specified alias/name within the provided certificate stores. The request and response objects will contain the fields shown in Table 634: POST /AddCert Request Body and Table 635: POST /AddCert Response Body.

Table 634: POST /AddCert Request Body

Parameter Name	Parameter Value
Keystores	Array of the certificate stores to which the provided entry should be added, with the same format as the response to GET /Keystores (see Table 649: GET /Keystores Response Body).
Alias	Name of the entry to which the certificate should be added. This parameter can also take a list of Certificate Store Type and Alias entries. If just a name is given, the certificate will have the same alias in all certificate stores it is added to. If a list is given, the certificate will have the same alias for each given store with the same certificate store type.
Overwrite	Boolean denoting if the entry should be overwritten, if one exists. An error will be returned if this is set to false, and an entry with the same alias/name exists.
Contents	PEM of the certificate to be added. This field is optional if a CertificateId is provided.
CertificateId	Database identifier within Keyfactor Command of the certificate to be added. This field is optional if the Contents are provided.

Table 635: POST /AddCert Response Body

Parameter Name	Parameter Value
Result	Numerical code indicating the result of the operation, as described in Table 639: POST /AddCertStoreServer Response Body.
Message	Description of the result of the operation, e.g. "The operation completed successfully".
InvalidKeystores	Array of certstores provided in the request for which the operation could not be completed. Entries will be formatted as follows:

Parameter Name	Parameter Value
KeystoreId	Guid of the certstore
ClientMachine	Machine hosting the certstore
StorePath	File path to the store on its machine
Alias	Alias for certificate to be added
Reason	Numerical code for the failure. Will take one of the following values:

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td></td><td>
<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Error Message</th></tr>
<tr><td>0</td><td>The certificate store was not found.</td></tr>
<tr><td>1</td><td>A job to add this certificate to this alias already exists.</td></tr>
<tr><td>2</td><td>No agent is available to perform this job.</td></tr>
</table>
</td></tr>
<tr><td>Explanation</td><td>A description of the failure encountered.</td></tr>
</table>
</td></tr>
</table>

Example Request

Multiple Alias entries

POST http://<host>/CMSApi/CertStore/1/AddCert HTTP/1.1

```
{
       "Keystores":
       [
       {"Id": "", "ClientMachine": "<client-machine>", "StorePath": "<store-path>"},
       {"Id": "", "ClientMachine": "<client-machine>", "StorePath": "<store-path>"}
       ],
       "Alias": {"<store type Id>":"<alias>","<store type Id":"alias"}
       "Overwrite": true,
       "CertificateId": "<certificate-id>",
       "Contents":  "-----BEGIN CERTIFICATE-----
       <base64-encoded-certificate-contents>
       -----END CERTIFICATE-----"
}
```

Example Request

String Alias

POST http://<host>/CMSApi/CertStore/1/AddCert HTTP/1.1

```
{
       "Keystores":
```

```
        [
        {"Id": "", "ClientMachine": "<client-machine>", "StorePath": "<store-path>"},
        {"Id": "", "ClientMachine": "<client-machine>", "StorePath": "<store-path>"}
        ],
        "Alias": "<alias>",
        "Overwrite":true,
        "CertificateId":"<certificate-id>",
        "Contents": "-----BEGIN CERTIFICATE-----
        <base64-encoded-certificate-contents>
        -----END CERTIFICATE-----"
}
```

Example Response

```
{
        "Result": 1,
        "Message" : "The operation completed successfully.",
        "InvalidKeystores": []
}
```

2.3.5.2 CertStore AddCertStore

The AddCertStore method allows a client to define a new certificate store within Keyfactor Command. The structure is as follows:

Table 636: POST /AddCertStore Request Body

Parameter Name	Parameter Value
StoreType	Type of certificate store to be defined. This field is required and allowed values are: **Parameter Name** / **Parameter Value** 0 — Java Keystore 2 — PEM file 3 — F5 SSL Profiles 4 — IIS Trusted Root Certificates

Parameter Name	Parameter Value
	<table><tr><th>Parameter Name</th><th>Parameter Value</th></tr><tr><td>5</td><td>NetScaler</td></tr><tr><td>6</td><td>IIS Personal Certificates</td></tr><tr><td>7</td><td>F5 Web Server</td></tr><tr><td>8</td><td>IIS Revoked Certificates</td></tr><tr><td>100</td><td>Amazon Web Services</td></tr><tr><td>101</td><td>File Transfer Protocol</td></tr></table>
ClientMachine	Machine where the certificate store resides (or will reside). Required.
StorePath	Path on the client machine where the store should be defined. Required for Java Keystore, PEM file, F5 SSL Profiles, and NetScaler (categories 0, 2, 3, and 5).
AgentId	Identifier of agent that will service the request. Either AgentId or AgentName must be provided for F5 (categories 3 and 7), IIS (categories 4, 6, and 8), and NetScaler stores (category 5).
AgentName	Machine name of agent that will service the request. Either AgentId or AgentName must be provided for F5 (categories 3 and 7), IIS (categories 4, 6, and 8), and NetScaler stores (category 5).
Container	Certificate store container that should contain the certificate store. This is optional and no certstore container will be assigned if it is not provided. See the *Keyfactor Command Reference Guide* for information on certificate store containers.
Password	Password used to access the store. Required for Java Keystore and optional for PEM file.
PrivateKeyPath	Path on the client machine where the private key should be stored. Supported only for PEM files, and is optional in that case. If no path is provided for a PEM file, the private key will be stored in the same PEM file as the certificate.

Table 637: POST /AddCertStore Response Body

Parameter Name	Parameter Value
Message	Description of the result of the operation, e.g. "The operation completed successfully".
Result	Numerical code for the outcome of the operation, as given in Table 639: POST /AddCertStoreServer Response Body.
Id	GUID of the created store, if successful.

Example Request

POST http://<host>/CMSApi/CertStore/1/AddCertStore HTTP/1.1

```
{
       "ClientMachine": "192.168.41.171",
       "StorePath": "/opt/cms-java-agent/config/trust.jks",
       "StoreType": 0,
       "Password": "changeit"
}
```

Example Response

```
{
       "Result": 1,
       "Message": "The operation completed successfully.",
       "Id": "b195c1f9-1957-4bdb-a15d-f45159482611"
}
```

2.3.5.3 CertStore AddCertStoreServer

Some certificate stores are managed by agents accessing the store through a third-party Web API. This currently includes F5 BigIP devices and Citrix NetScaler devices. These stores require the definition of a certstore server before the store itself can be defined in Keyfactor Command. Each server can be configured with a location and user credentials to access the client machine via the appropriate third-party API. This Keyfactor Command Web API method allows such configuration. The structure shown in Table 638: POST /AddCertStoreServer Request Body should be used for requests.

Table 638: POST /AddCertStoreServer Request Body

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td>Name</td><td>Hostname of the machine the agent will connect to.</td></tr>
<tr><td>ServerType</td><td>Platform for this server, defining what certstore types are supported. Allowed values are:
<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td>0</td><td>F5</td></tr>
<tr><td>1</td><td>NetScaler</td></tr>
</table>
</td></tr>
<tr><td>UseSSL</td><td>Boolean denoting whether the agent should connect to the client API using https or http.</td></tr>
</table>

Parameter Name	Parameter Value
Username	Username to provide to the client API.
Password	Password corresponding to the login for the given Username to access the client API.

Table 639: POST /AddCertStoreServer Response Body

Parameter Name	Parameter Value
Result	Status code for the operation. Will take one of the following values: **Value** / **Description** 1 / Success 2 / Failure 3 / Warning
Message	Description of the operation outcome, e.g. "The operation completed successfully".

Example Request

POST http://<host>/CMSApi/CertStore/1/AddCertStoreServer HTTP/1.1

```
{
       "Name": "192.168.23.100",
       "UseSSL" : true,
       "Username": "nsroot",
       "Password": "nsroot",
       "ServerType": 1
}
```

Example Response

```
{
       "Result": 1,
       "Message": "The operation completed successfully."
}
```

2.3.5.4 CertStore AddCertStoreType

The POST /AddCertStoreType method will create a certificate store type that will be used for a custom certificate store that extends the Keyfactor Command Agent's Any Agent functionality. The parameters that can be used for this endpoint are shown in Table 640: POST /AddCertStoreType Request Body, while the response format can be found in Table 641: POST /AddCertStoreType Response Body.

Table 640: POST /AddCertStoreType Request Body

Parameter Name	Parameter Value
Name	The name the certificate store type will have in Keyfactor Command. This parameter is **required**.
ShortName	The short name of the certificate store type. This parameter is **required**.
AddSupported	A Boolean that sets if the certificate store of this certificate store type is allowed to be added to. This parameter is **required**.
CreateSupported	A Boolean that sets if the certificate store of this certificate store type is allowed to be created if missing. This parameter is **required**.
DiscoverySupported	A Boolean that sets if the certificate store of this certificate store type is allowed to be discovered in a discovery scan. This parameter is **required**.
RemoveSupported	A Boolean that sets if the certificate store of this certificate store type allows certificates to be removed from it. This parameter is **required**.
EnrollmentSupported	A Boolean that sets if the certificate store of this certificate store type supports reenrollment. This parameter is **required**.
EntryPasswordSupported	A Boolean that sets if the certificate store of this certificate store type supports an entry password. This parameter is **required**.
PrivateKeyAllowed	A parameter that sets requirements on the private key of a certificate being entered into the certificate store. This parameter is **required**. Valid values are: **Value** / **Name** 0 / Forbidden 1 / Optional 2 / Required
LocalStore	A Boolean that sets if the certificate store of this certificate store type requires a certificate store server. This parameter is **required**.
StorePasswordRequired	A Boolean that sets if the certificate store of this type requires a password. This para-

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td></td><td>meter is required.</td></tr>
<tr><td>StorePathType</td><td>The type used for the certificate store path.<table><tr><th>Option</th><th>Description</th></tr><tr><td>Empty</td><td>Path will be a free form field.</td></tr><tr><td>String</td><td>Path will only be the specified string.</td></tr><tr><td>Comma Separated String</td><td>Path will need to be chosen from the list given.</td></tr></table></td></tr>
<tr><td>CustomAliasAllowed</td><td>A Boolean that sets whether the certificate store of this type allows a custom alias. This parameter is optional.</td></tr>
<tr><td>Powershell</td><td>A Boolean that sets whether the certificate store of this type uses PowerShell. This parameter is optional.</td></tr>
<tr><td>ServerRegistration</td><td>A Boolean that sets whether Keyfactor Command needs to prompt for credentials for each client machine that has that certificate store type. This parameter is optional.</td></tr>
<tr><td>JobProperties</td><td>A comma separated string defining properties that are required when performing management jobs on a certificate store of this type. This parameter is optional.</td></tr>
<tr><td>Properties</td><td>A dictionary of any extra properties a certificate store of this type would need. This parameter is optional. If this property is provided, a type is required. Parameters of a property are:<table><tr><th>Field</th><th>Description</th></tr><tr><td>DisplayName</td><td>The name of the property. This parameter is optional.</td></tr><tr><td>Type</td><td>The type of the property. This parameter is required. Valid values are: String, Bool, MC, and Secret</td></tr><tr><td>Required</td><td>A Boolean that sets whether the property is required in the certificate store.</td></tr><tr><td>Depends</td><td>If this is not the first property, this property can depend on another property. The property name is used to determine which property is being depended on.</td></tr><tr><td>Value</td><td>A default Value of the property.</td></tr></table></td></tr>
</table>

Table 641: POST /AddCertStoreType Response Body

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td>Message</td><td>Description of the operation outcome, e.g. "The operation completed successfully".</td></tr>
<tr><td>Result</td><td>Status code for the operation. Will take one of the following values:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>Success</td></tr>
<tr><td>2</td><td>Failure</td></tr>
<tr><td>3</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>Data</td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Name</td><td>The name of the type.</td></tr>
<tr><td>ShortName</td><td>The ShortName of the type.</td></tr>
<tr><td>StoreType</td><td>The Id of the store</td></tr>
<tr><td>LocalStore</td><td>A Boolean if the certificate store is on the local server of the agent.</td></tr>
<tr><td>ServerRegistration</td><td>Tells whether server registration is needed by Keyfactor Command.</td></tr>
<tr><td>ImportType</td><td>A value to indicate the source of a certificate record in the Keyfactor Command audit logs.</td></tr>
<tr><td>InventoryJobType</td><td>The GUID of the inventory job type that is used to register with the Any Agent.</td></tr>
<tr><td>ManagementJobType</td><td>The GUID of the management job type that is used to register with the Any Agent.</td></tr>
<tr><td>AddSupported</td><td>A Boolean stating whether an add job will be supported by the certificate store.</td></tr>
<tr><td>RemoveSupported</td><td>A Boolean stating whether a remove job will be supported by the certificate store.</td></tr>
<tr><td>CreateSupported</td><td>A Boolean stating whether a create job will be supported by the certificate store.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>DiscoverySupported</td><td>A Boolean stating whether a discovery job will be supported by the certificate store.</td></tr>
<tr><td>EnrollmentSupported</td><td>A Boolean stating whether an enrollment job will be supported by the certificate store.</td></tr>
<tr><td>InventoryEndpoint</td><td>The endpoint that will be hit by the agent.</td></tr>
<tr><td>Properties</td><td>A list of properties that reflect those given in the request.</td></tr>
<tr><td>EntryPasswordSupported</td><td>A Boolean stating whether an entry password will be supported by the certificate store.</td></tr>
<tr><td>StorePasswordRequired</td><td>A Boolean stating whether a store password will be required by the certificate store.</td></tr>
<tr><td>PrivatekeyAllowed</td><td>An integer notifying the state of the private keys in the certificate store.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0</td><td>Forbidden</td></tr>
<tr><td>1</td><td>Optional</td></tr>
<tr><td>2</td><td>Required</td></tr>
</table>
</td></tr>
<tr><td>StorePathType</td><td>The value of the store path. If value is an empty string, the field is free form.</td></tr>
<tr><td>CustomAliasAllowed</td><td>A Boolean stating whether a custom alias will be supported by the certificate store.</td></tr>
<tr><td>JobProperties</td><td>The properties that will be required when performing a management job on the certificate store with this type.</td></tr>
</table>
</td></tr>
</table>

Example Request

POST http://<host>/CMSApi/CertStore/1/AddCertStoreType HTTP/1.1

```
{
       "Name": "<Type Name>",
       "ShortName": "<Type Short Name>",
       "AddSupported": true,
       "CreateSupported": false,
       "DiscoverySupported": true,
       "RemoveSupported": true,
       "EnrollmentSupported": true,
       "EntryPasswordSupported": true,
       "PrivateKeyAllowed": <integer 0-2>,
       "LocalStore": true,
       "StorePasswordRequired": true,
       "Powershell":false,
       "CustomAliasAllowed":false,
       "JobProperties":"<List of Job Properties>",
       "ServerRegistration": false,
       "Properties": {
              “<Property Name>": {
                     "type":"<Property Type>",
                     “DisplayName”: “<Discplay Name>”
              },
              "<Property Name>":{
                     "type":"<Type>",
                     “displayName”:”<Display Name>”
                     "value": "<Value>"
              }
       },
       "StorePathType": <Path Type>
}
```

Example Response

Status Code: 200

```
{
       "Message": "The operation completed successfully.”,
       "Result": 1,
       "Data": {
              "Name": "<Name>",
              "ShortName": "<Short Name>",
              "StoreType": <Store Type Id>,
              "LocalStore": true,
              "ServerRegistration": null,
              "ImportType": <Import Type>,
```

```
            "InventoryJobType": "<Inventory Job Type Guid>",
            "ManagementJobType": "<Management Job Type Guid>",
            "AddSupported": false,
            "RemoveSupported": true,
            "CreateSupported": false,
            "DiscoveryJobType": "<Discovery Job Type Guid>",
            "EnrollmentJobType": "<Enrollment Job Type Guid>",
            "InventoryEndpoint": “<Inventory Endpoint>",
            "Properties": {
                    "<Property Name>": {
                            "Type": "<Type>",
                            "DisplayName": "<Discplay Name>",
                            "Required": false,
                            "Depends": null,
                            "Value": <Value>
                    },
                    "<Property Name>": {
                            "Type": "<Type>",
                            "DisplayName": “<Display Name> ",
                            "Required": false,
                            "Depends": null,
                            "Value": "<Value>"
                     }
            },
            "EntryPasswordSupported": true,
            "StorePasswordRequired": true,
            "PrivateKeyAllowed": <Integer 0-2>,
            "StorePathType": <Store Path Type>,
            "CustomAliasAllowed": false,
            "JobProperties": "<Job Properties>"
    }
}
```

2.3.5.5 CertStore AddPFX

The POST AddPfx method will schedule the addition of the provided PFX(s) to the specified alias/name within the provided certificate store(s). The request should contain the fields shown in Table 642: POST /AddPfx Request Body, while the response format will be the same as for AddCert (see Table 635: POST /AddCert Response Body).

Table 642: POST /AddPfx Request Body

Parameter Name	Parameter Value
Keystores	Array of certificate stores to which the provided entry should be added, with the same format as the response to GET /Keystores (see Table 649: GET /Keystores Response Body).

Parameter Name	Parameter Value
Alias	Name of the entry to which the certificate should be added.
Overwrite	Boolean denoting if the entry should be overwritten, if one exists. An error will be returned if this is set to false but an entry with the same alias/name exists.
Contents	PEM of the PFX to be added. Do not include the …BEGIN… AND …END… lines.
PfxPassword	Password of the PFX.
HasEntryPassword	Boolean denoting if the password required for the entry is different than that of the certificate store itself.
EntryPassword	Password for the certificate store entry. Required if the HasEntryPassword is set to true.

Example Request

POST http://<host>/CMSApi/CertStore/1/AddPfx HTTP/1.1

```
{
       "Keystores":
       [{
       "Id": "<keystore-id>",
       "ClientMachine": "<client-machine>",
       "StorePath": "<store-path>"
       },
       {
       "Id": "<keystore-id>",
       "ClientMachine": "<client-machine>",
       "StorePath": "<store-path>"
       }],
       "Alias": "<alias>",
       "Overwrite": "true",
       "HasEntryPassword": "true",
       "EntryPassword": "<entry-password>",
       "PfxPassword": "<pfx-password>",
       "Contents": "<base64-encoded PFX>"
}
```

2.3.5.6 CertStore CreateJKS

In most cases, certificate stores will already exist on the client machine prior to configuration within Keyfactor Command. For example, the IIS Personal Store exists on each windows machine independently of Keyfactor Command installation. In other cases, such as PEM files, the file can be created when a certificate is added. However, with a Java Keystore, creating the store on the file system and adding certificates to it are different

operations. The CreateJKS method supports scheduling creation of a Java Keystore as a Keyfactor Command Agent job. The structure of this request is given in Table 643: POST /CreateJKS Request Body while the response is the same as for AddCertStore (see Table 637: POST /AddCertStore Response Body).

Table 643: POST /CreateJKS Request Body

Parameter Name	Parameter Value
ClientMachine	Machine on which the certificate store will reside.
StorePath	Path and filename of the certificate store to be created.
Password	Password to use for the new store.

Example Request

POST http://<host>/CMSApi/CertStore/1/CreateJKS HTTP/1.1

```
{
       "ClientMachine" : "192.168.41.171",
       "StorePath" : "/opt/cms-java-agent/config/trust.jks",
       "Password" : "changeit"
}
```

2.3.5.7 CertStore EditCertStore

The EditCertStore method allows certain aspects of a cert store definition to be updated. Some aspects, such as the store type and client machine, cannot be updated. The format of the request given in Table 644: POST /EditCertStore Request Body, while the response will be as it is for Table 639: POST /AddCertStoreServer Response Body.

Table 644: POST /EditCertStore Request Body

Parameter Name	Parameter Value
Id	Guid – Unique identifier of the certificate store. This field is the most specific, and does not require either the ClientMachine or StorePath fields to be provided.
ClientMachine	Machine on which the store resides. This field is required if the Id field is not provided.
StorePath	Path and filename of the certificate store. This field is required if the Id field is not provided.
NewStorePath	New path on the machine filesystem where the certstore resides.
NewContainer	Reassign the certstore container in Keyfactor Command where this store is configured.
NewPassword	Change the password used by the agent to access the store.

Parameter Name	Parameter Value
NewPrivateKeyPath	Change the path of a private key stored separately from a PEM file certificate
NewAgentId	Change the agent managing a remote certstore by providing its GUID. Cannot be used with NewAgentName.
NewAgentName	Change the agent managing a remote certstore by providing the name it reports to Keyfactor Command. Cannot be used with NewAgentId.

Example Request

POST http://<host>/CMSApi/CertStore/1/EditCertStore HTTP/1.1

```
{
       "ClientMachine": "192.168.23.100",
       "StorePath" : "/nsconfig/ssl",
       "NewStorePath" : "/nsconfig/ssl/vserver1",
       "NewContainer": "NetScaler"
}
```

2.3.5.8 CertStore EditCertStoreServer

A cert store server is a machine that hosts a store that is remotely managed by a Keyfactor Command Agent, such as a NetScaler or F5 device. The CertStoreServer configuration contains the data that allows the agent to connect to the host via the host platform's API. This method allows configuration of an existing CertStoreServer to be updated. The request format is shown in Table 645: POST /EditCertStoreServer Request Body, while the response format is the same as for AddCertStoreServer (see Table 639: POST /AddCertStoreServer Response Body).

Table 645: POST /EditCertStoreServer Request Body

Parameter Name	Parameter Value
Name	Hostname of the machine the agent will connect to. Required if Id is not provided.
Id	Identifier of the certstore server to update. Required if Name is not provided.
UseSSL	Boolean denoting whether the agent should connect to the client API using https or http.
NewUsername	Username to provide to the client API. Required if NewPassword is provided.
NewPassword	Password corresponding to the login for the given Username to access the client API. Required if NewUsername is provided.

Example Request

POST http://<host>/CMSApi/CertStore/1/EditCertStoreServer HTTP/1.1

```
{
       "Name": "192.168.23.100",
       "UseSSL" : true,
       "newUsername" : "myNetScalerAdmin",
       "newPassword": "S1deways-Grassh0pper4979"
}
```

2.3.5.9 CertStore GetCertStoreTypes

The GET CertStoreTypes method returns a list of all certificate store types. The format for each element in the list can be found in Table 646: GET /GetCertStoreTypes Response Body.

Table 646: GET /GetCertStoreTypes Response Body

Parameter Name	Parameter Value
Name	The name of the type.
ShortName	The short name of the type.
StoreType	The Id of the type.
LocalServer	A Boolean stating if the certificate store server is the same machine as the agent.
ServerRegistration	A Boolean stating whether Keyfactor Command needs to prompt for credentials for each client machine that has this certificate store type.
InventoryJobType	The GUID of the Inventory Job.
ManagementJobType	The GUID of the management job.
DiscoveryJobType	The GUID of the discovery job.
EnrollmentJobType	The GUID of the enrollment job.
InventoryEndpoint	The server endpoint to which the agent publishes its inventory results.
Properties	The added properties of the certificate store that uses this type.
EntryPasswordSupported	A Boolean stating if an entry password is supported by the certificate store that uses this type.
StorePasswordRequired	A Boolean stating if a store password is required by the certificate store that uses this type.

Parameter Name	Parameter Value
PrivateKeyAllowed	A Boolean stating if a private key is allowed by the certificate store that uses this type.
StorePathType	The value for the store path. Can be null, a string or a comma-separated string for free form, the specified path or a list of paths to choose from respectively.

2.3.5.10 CertStore Inventory

The POST Inventory method returns a list of the entries within the provided certificate store. The request body is formatted the same as the response to GET /Keystores (see Table 649: GET /Keystores Response Body).

Table 647: POST /Inventory Response Body

Parameter Name	Parameter Value
Alias	Alias/name of the certificate store entry.
PrivateKeyEntry	Boolean value denoting if the entry has an associated private key.
Certificates	Array of the certificates contained within the certificate store (see Table 648: POST /Inventory Response Certificates Fields).

Table 648: POST /Inventory Response Certificates Fields

Parameter Name	Parameter Value
ChainLevel	Position of the certificate within the chain. This is only applicable for private key entries.
CertificateId	Database identifier of the certificate within Keyfactor Command.
Thumbprint	Thumbprint of the certificate.

Example Request

POST http://<host>/CMSApi/CertStore/1/Inventory HTTP/1.1

```
{
       "Id": "<certificate-store-id>",
       "ClientMachine": "<client-machine>",
       "StorePath": "<store-path>"
}
```

Example Response

```
[
       {
       "Alias": "<alias1>",
       "PrivateKeyEntry": false,
       "Certificates": [{"ChainLevel":0,"CertificateId":<id>,"Thumbprint":"<thumbprint>"}]
       },
       {
       "Alias": "<alias2>",
       "PrivateKeyEntry": true,
       "Certificates":
       [
       {"ChainLevel": 0,"CertificateId": <id>,"Thumbprint": "<thumbprint>"},
       {"ChainLevel": 1,"CertificateId": <id>,"Thumbprint": "<thumbprint>"},
       {"ChainLevel": 2,"CertificateId": <id>,"Thumbprint": "<thumbprint>"}
       ]
       }
]
```

2.3.5.11 CertStore Keystores

The GET Keystores method returns a list of the certificate stores within Keyfactor Command. This method requires no parameters. An array of the certificate stores is returned. The information shown in Table 649: GET /Keystores Response Body is returned for each certificate store in the array.

Table 649: GET /Keystores Response Body

Parameter Name	Parameter Value
Id	The Keyfactor Command request database identifier of the certificate store.
ClientMachine	Host name of the machine on which the certificate store resides.
StorePath	Path or other identifier of the certificate store (e.g. "IIS Personal" for IIS Personal stores).

Example Request

GET http://<host>/CMSApi/Certstore/1/Keystores

Example Response

```
[
       {
       "Id": "<certificate-store-id>",
       "ClientMachine": "<client-machine>",
```

```
        "StorePath": "<store-path>"
        },
        {
        "Id": "<certificate-store-id>",
        "ClientMachine": "<client-machine>",
        "StorePath": "<store-path>"
        }
]
```

2.3.5.12 CertStore Remove

The POST Remove method will schedule the removal of the provided entry associated with the specified alias/-name within the provided certificate store(s). The request should contain the fields shown in Table 650: POST /Remove Request Body, while the response will be formatted as it is for AddCert and AddPfx (see Table 635: POST /AddCert Response Body).

Table 650: POST /Remove Request Body

Parameter Name	Parameter Value
Keystores	Array of the certificate stores from which the provided entry should be removed, formatted as with the GET /Keystores response (see Table 649: GET /Keystores Response Body).
Alias	Name of the entry from which the certificate should be removed.
Thumbprint	Thumbprint of the certificate to be removed. This field is optional if the CertificateId is provided.
CertificateId	Database identifier within Keyfactor Command of the certificate to be removed. This field is optional if the Thumbprint is provided.

Example Request

POST http://<host>/CMSApi/CertStore/1/Remove HTTP/1.1

```
{
        "Keystores":
        [{
        "Id": "<keystore-id>",
        "ClientMachine": "<client-machine>",
        "StorePath": "<store-path>"
        },
        {
        "Id": "<keystore-id>",
```

```
        "ClientMachine": "<client-machine>",
        "StorePath": "<store-path>"
        }],
        "Alias": "<alias>",
        "CertificateId": "<certificate-id>",
        "Thumbprint": "<thumbprint>"
}
```

2.3.5.13 CertStore ScheduleInventory

Keyfactor Command Agents typically monitor the contents of cert stores they manage on a pre-configured interval, either once per day or every n minutes. The ScheduleInventory endpoint allows this interval configuration to be updated or switched on and off. Requests are formatted as follows, while the response is formatted as for AddCertStoreServer (see Table 639: POST /AddCertStoreServer Response Body):

Table 651: POST /ScheduleInventory Request Body

Parameter Name	Parameter Value
Id	Guid – Unique identifier of the certificate store. This field is the most specific, and does not require either the ClientMachine or StorePath fields to be provided.
ClientMachine	Machine on which the certificate store resides. This field is optional if the Id field is provided. It is required if used in conjunction with the ClientMachine field.
StorePath	Path and filename of the certificate store. This field is optional if the Id field is provided. It is required if used in conjunction with the ClientMachine field.
ScheduleType	Value indicating whether inventory should be off, on an interval, or daily. Possible values are: **Parameter Name** / **Parameter Value** 0 – Off 1 – Interval 2 – Daily
ScheduleTime	Time of day (hour and minute) that the inventory should run. Used for ScheduleType "Daily".
ScheduleInterval	Integer number of minutes that should elapse between inventories. Used for ScheduleType "Interval".
Overwrite	Boolean indicating whether a previous schedule configuration, if it exists, should be overwritten with the provided schedule configuration.

Example Request

POST http://<host>/CMSApi/CertStore/1/ScheduleInventory HTTP/1.1

```
{
       "ID": "832f87c7-0af7-4043-9840-3022faeeae45",
       "ClientMachine": "192.168.41.171",
       "StorePath": "/home/pi/cherry/cherrystore",
       "ScheduleType": 2,
       "ScheduleTime: "23:00",
       "Overwrite": true
}
```

Example Response

Status Code: 200

```
{
       "Message": "The operation completed successfully.",
       "Result": 1
}
```

2.3.6 Metadata

Metadata in Keyfactor Command allows dynamic information about a certificate, or other data associated with a certificate that isn't included in the cert itself, to be associated with the certificate within Keyfactor Command. A metadata field can be defined within Keyfactor Command of a given type with a variety of other attributes, such as default values and security constraints. Currently, the supported metadata types are:

- String
  Alphanumeric text field limited to 400 characters.
- Integer
  Supports whole numbers only.
- Date
- Multiple Choice
- Big Text
  Big Text fields are limited to 4000 characters. String fields support additional indexing, and so may be preferable to Big Text fields for large databases where possible.
- Boolean
  True/False

Every certificate in Keyfactor Command can be assigned a value for each metadata field defined. The Metadata Web API component supports assignment, retrieval, and comparison of metadata values associated with a certificate, as well as retrieval of metadata field definitions. The supported methods are listed in Table 652: Metadata Endpoints.

NOTE: Since the Certificates/1/Metafield endpoint (see Certificates Metafield on page 1413) is considered "Version 1" of this API component, version numbering here starts at 2.

Table 652: Metadata Endpoints

Endpoint	Method	Description
/2/Compare	POST	Compare the stored value for a metadata field associated with a given certificate against the value given in the request, and return a Boolean indicating whether the values match.
/2/Get	POST	Return the value for a metadata field associated with a given certificate.
/2/Set	POST	Assign a value for a metadata field to a given certificate.
/3/Get	POST	Return the value for a metadata field associated with a given certificate.
/3/GetDefinition	POST	Return the definition for the metadata field with given name.
/3/Set	POST	Assign a value for a metadata field to a given certificate.

2.3.6.1 Metadata V2

The *Metadata/2/...* calls all have a common request format and set of response codes. The request body is always a JSON-formatted string containing a set of fields used to identify the certificate the operation is to be performed on and a list of key-value pairs defining the metadata fields of interest. In the case of the Set method, the values to which each field should be set must also be provided.

Table 653: POST Metadata/2/ Request Body*

Parameter Name	Parameter Value
Key	The key value can either be "Thumbprint" or "Serial" to identify the certificate. If you choose serial, you must include both the SerialNumber and the IssuerDN fields.
SerialNumber	The serial number of the certificate. Required only if Key is set to "Serial".
IssuerDN	The issuer of the certificate. Required only if Key is set to "Serial".
Thumbprint	The thumbprint of the certificate. Required only if Key is set to "Thumbprint".
metadatalist	Metadata field/value entries in one of the following forms: • [{"MetadataFieldTypeName" : "<field1-name>", "Value" : "<field1-value>"}, {...}, {...}] • {"<field1-name>" : "<field1-value>", "<field2-name>" : "<field2-value>"}

Metadata V2 Set

The Metadata V2 Set POST method is used to set metadata value on a certificate in Keyfactor Command. The POST request body must consist of a JSON string containing the parameters used to set a certificate's metadata. If the request is successful, a 200 OK will be returned with "true" in the message body. If it is not, an appropriate 4xx HTTP status code is returned, and the body will contain a JSON object with a message about the error.

Example Request

Using thumbprint

POST http://<host>/CMSApi/Metadata/2/Set

```
{
       "Key": "Thumbprint",
       "Thumbprint": "<thumbprint>",
       "metadatalist" : [{"EmailAddress":"bob.smith@example.com"}]
}
```

Example Request

Using serial

POST http://<host>/CMSApi/Metadata/2/Set HTTP/1.1

```
{
       "Key": "Serial",
       "SerialNumber": "<serial-number>",
       "SerialIssuer": "<issuing-ca>",
       "metadatalist": [{"EmailAddress": "bob.smith@example.com"}]
}
```

Example Response

(Unsuccessful)

```
{
       "Message": "The following metadata errors were found: 'myInvalidField' was not a valid
MetadataFieldTypeName."
}
```

Metadata V2 Get

The Metadata V2 Get POST method is used to get metadata value on a certificate in Keyfactor Command. Despite the "Get" in the Keyfactor Command method name, the HTTP method must be POST and not GET. As with metadata/2/set (see Metadata V2 Set on the previous page), the POST request body must consist of a JSON string containing the parameters used to get a certificate's metadata. The "value" attribute for each entry in the metadatalist is not used, but must be present and can be set to null or an empty string.

Example Request

Using thumbprint

POST http://<host>/CMSApi/Metadata/2/Get HTTP/1.1

```
{
       "Key": "Thumbprint",
       "Thumbprint": "<thumbprint>",
       "metadatalist": [{"MetadataFieldTypeName": "EmailAddress", "Value": ""}]
}
```

Example Response

```
{
       "EmailAddress" : "bob.smith@example.com"
}
```

Metadata V2 Compare

The Metadata V2 Compare method takes a collection of metadata and returns a true/false response depending on whether the values for the fields provided match the values stored in Keyfactor Command. 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 third-party applications.

Example Request

POST http://<host>/CMSApi/Metadata/2/Compare HTTP/1.1

```
{
       "Key": "Thumbprint",
       "Thumbprint": <Thumbprint>
       "metadatalist": [{"MetadataFieldTypeName": "EmailAddress",
```

```
      "Value": "example@example.com"}]
}
```

2.3.6.2 Metadata V3

Version 3 of the metadata API allows more flexibility in certificate lookup and security measures than version 2, while allowing more to be done in a single API call and with a more concise JSON representation. Requests to metadata v3 API methods include 3 parts as shown in Table 654: Metadata V3 Request Body.

Table 654: Metadata V3 Request Body

Parameter Name	Parameter Value
Lookup	Given in Table 4: Classic API Certificate Lookup Structure.
Security	Given in Table 655: Metadata V3 Security Bitflags
Metadata	Dictionary of key-value pairs, where the key represents the metadata field and (for the set method) the value represents the value to be associated to the certificate referenced in the "Lookup" value. For Get and GetDefinition methods, the same structure is used but the value is not considered.

The security parameter includes a set of required flags, certain of which necessitate the inclusion of other parameters. The flags should be passed as integers, combined together using bitwise OR. The flags defined in Keyfactor Command are described in Table 655: Metadata V3 Security Bitflags.

Table 655: Metadata V3 Security Bitflags

Value	Definition
00000001	Fail if certificate has been revoked or denied.
00000010	Fail if certificate has expired.
00000100	Fail if certificate status is pending or unknown.
00001000	Fail if metadata values provided for authentication do not match the values stored in Keyfactor Command. Must be paired with an "authmetadata" field, the value of which is a dictionary formatted with {"MetadataFieldName" : "AssociatedCertificateValue" pairs}. This effectively supplants the "Compare" method found in v2.
00100000	Overwrite flag – update value even if field is configured to require explicit overwrites and a value has been associated with the certificate (applies to Set method only).

The metadata argument is a JSON dictionary containing 0 or more key-value pairs. In each pair, the key must correspond to the name of a metadata field. The value, if present, must be of a data type matching the type of the

field. For Boolean and integer metadata field values, this is the JSON Boolean or integer type, respectively, while all other metadata field types are to be represented as strings. Dates should be passed in the "YYYY-M-D" format. Multi-valued entries should have a value that exactly matches one of the pre-defined values. For the Get method, values need not be provided and the empty string can be used as the value for each key. In the case where there are 0 metadata arguments, the "Metadata" key must still be present and mapped to an empty object "{}". Note that this syntax is different than previous Metadata API versions, and uses a more concise format. An example is:

```
"Metadata" : {"Email-Contact" : "user@example.com", "Contact-Name" : "John Doe", "ID-number" : 738}
```

Metadata V3 Set

The Metadata V3 Set POST method is used to set metadata value on a certificate in Keyfactor Command. It returns a "200 OK" response with no further content on success.

Example Request

Using thumbprint

POST http://<host>/CMSApi/Metadata/3/Set HTTP/1.1

```
{
       "Lookup":
       {
              "Type": "Thumbprint",
              "Thumbprint": "<thumbprint>"
       },
       "Security": {"Flags": 3},
       "Metadata": {"Email-Contact": "bob.smith@example.com"}
}
```

Example Request

Using serial

POST http://<host>/CMSApi/Metadata/3/Set HTTP/1.1

```
{
       "Lookup":
       {
              "Type": "Serial",
              "SerialNumber": "<serial-number>",
              "IssuerDN": "<issuer-dn>"
       },
```

```
        "Security": {"Flags" : 3},
        "Metadata": {"Email-Contact": "bob.smith@example.com"}
}
```

Metadata V3 Get

The Metadata V3 Get POST method is used to get metadata value on a certificate in Keyfactor Command. Despite the "Get" in the Keyfactor Command method name, the HTTP method must be POST and not GET. As with metadata/3/set (see Metadata V3 Set on the previous page), the POST request body must consist of a JSON string containing the parameters used to get a certificate's metadata. The "value" attribute for each metadata entry is not used, but must be present and can be set to null or an empty string. The method returns a JSON dictionary in a format identical to the metadata parameter, with key-value pairs containing the fields and values requested.

Example Request

POST http://<host>/CMSApi/Metadata/3/Get HTTP/1.1

```
{
        "Lookup":
        {
                "Type": "Serial",
                "SerialNumber": "<serial-number>",
                "IssuerDN": "<issuer-dn>"
        },
        "Security": {"Flags": 3},
        "Metadata ": {"Email-Contact": ""}
}
```

Example Response

```
{
        "Email-Contact": "bob.smith@example.com"
}
```

Metadata V3 GetDefinition

The Metadata V3 GetDefinition API endpoint will return the definition of a metadata field. Note that, while this does not operate on a certificate, the same request structure is used so the fields must be supplied, but the value will not be used. The structure of the response is given below.

Table 656: POST /GetDefinition Response Body

Parameter Name	Parameter Value
Name	Name of the metadata field.
Description	Purpose or intended usage of the field.
Hint	Sample value to be shown when users enter a value for this field in the Keyfactor Command Management Portal.
Validation	Regular Expression string capturing acceptable values for this field.
Required	Boolean indicating whether certificates added to Keyfactor Command must include a value for this field.
Message	Error message to be returned for values that do not conform to the regular expression.
Options	Comma-separated list of allowed values for "multi-valued" metadata fields.
DefaultValue	Initial value to be assigned for new certificates if a value is not provided at addition time.
AllowAPI	Boolean indicating whether values for this field are exposed through API Get and Set requests.
ExplicitUpdate	Boolean indicating whether updates require an appropriate flag to overwrite previous values.

Example Request

POST http://<host>/CMSApi/Metadata/3/GetDefinition HTTP/1.1

```
{
       "Lookup": { "Type": "CMSID", "CMSID" : 1},
       "Security": {"Flags": 0},
       "Metadata ": {"Email-Contact": ""}
}
```

Example Response

```
{
       "Name": "Email-Contact",
       "Description": "Email contact for the certificate.",
       "Hint": "contact@domain.com",
       "Validation": null,
       "Required": false,
       "Message": null,
```

```
        "Options": null,
        "DefaultValue": null,
        "AllowAPI": true,
        "ExplicitUpdate": true
}
```

2.3.7 Security

The Security component of the Keyfactor Web APIs includes all methods necessary to programmatically add, get and delete security identities as well as get, add, edit and delete the security roles defined in Keyfactor Command. The complete set of methods in the component is given in 2.3.7 Security.

Table 657: Security Endpoints

Endpoint	Method	Description
/1/GetIdentities	GET	Return a list of the identities in Keyfactor Command, the roles they are assigned to and their validity
/1/AddIdentity	POST	Add an identity to Keyfactor Command
/1/DeleteIdentities	POST	Remove an identity from Keyfactor Command
/1/GetRoles	GET	Retrieve all the security roles currently defined in Keyfactor Command with all of their permissions, a description and who they are assigned to
/1/AddRole	POST	Add a security role to Keyfactor Command
/1/EditRole	POST	Edit a security role in Keyfactor Command
/1/DeleteRole	POST	Delete a security role from Keyfactor Command

2.3.7.1 Security GetIdentities

The GET GetIdentities request returns a list of identities known to Keyfactor Command with the type of identity (user or group), whether the identity is valid or not and the roles associated with the identity. No parameters or extra headers are necessary for this method.

Example Request

GET http://<host>/CMSApi/Security/1/GetIdentities HTTP/1.1

Example Response

Status Code: 200

```
[
	{
		"Id": <Id>,
		"AccountName": "<Domain>\\<Identity>",
		"Type": "<Identity Type>",
		"Roles": "<List of Roles>",
		"Valid": true
	}
]
```

2.3.7.2 Security AddIdentity

The POST AddIdentities request adds an identity to Keyfactor Command. The POST request must contain a JSON string containing the AD account name. This method returns a 200 with the Id, account name, type, roles, and validity of the identity. The request parameters can be found in Table 658: POST AddIdentity Request Parameter.

Table 658: POST AddIdentity Request Parameter

Parameter Name	Parameter Value
Account	The name of the account that is to be added to CMS. This parameter is required.

Example Request

For a user

POST http://<host>/CMSApi/Security/1/AddIdentity HTTP/1.1

```
{
	"Account": "<Domain>\\<User>"
}
```

Example Request

For a group

POST http://<host>/CMSApi/Security/1/AddIdentity HTTP/1.1

```
{
	"Account": "<Domain>\\<Group>"
}
```

Example Response

Status Code: 200

```
{
       "Id": <Id>,
       "AccountName": "<Domain>\\<Identity>",
       "Type": "<Identity Type>",
       "Roles": "<List of Roles>",
       "Valid": true
}
```

2.3.7.3 Security DeleteIdentity

The POST AddIdentities request removes an identity from Keyfactor Command. The POST request must contain a JSON string containing the identity Id. This method returns a 200 a message stating the identity was deleted successfully. The request parameters can be found in Table 659: POST DeleteIdentity Request Parameter

Table 659: POST DeleteIdentity Request Parameter

Parameter Name	Parameter Value
Id	The Id of the identity that is to be deleted

Example Request

POST http://<host>/CMSApi/Security/1/DeleteIdentity HTTP/1.1

```
{
       "Id": <Id>
}
```

Example Response

Status Code: 200

```
{
       "Message": "ADIdentity deleted successfully"
}
```

2.3.7.4 Security GetRoles

The GET GetRoles endpoint retrieves all the current security roles defined in Keyfactor Command and returns the Id, name, description, validity, permissions and associated identities. The response parameters can be found in

Table 660: POST /GetRoles Response Body.

Table 660: POST /GetRoles Response Body

Parameter Name	Parameter Value
Id	The Id of the security role.
Name	The name of the security role.
Description	The description of the security role.
Valid	The validity of the security role.
Permissions	The permissions of the security role.
Identities	The security identities of the security role.

Example Request

GET http://<host>/CMSApi/Security/1/GetRoles HTTP/1.1

Example Response

Status Code: 200

```
[
	{
		"Id": <Id>,
		"Name": "<Name>",
		"Description": "<Description>",
		"Valid": true,
		"Permissions": "<List of Permissions>",
		"Identities": "<List of Identities>"
	},
]
```

2.3.7.5 Security AddRole

The POST AddRole endpoint creates a security role in Keyfactor Command. This endpoint can be used to assign a role to an identity and permissions to a role.

The list of available permissions can be found in Table 661: Keyfactor Command Permissions List.

Request parameters can be found in Table 662: POST /AddRole Request Parameters.

Response parameters can be found in Table 660: POST /GetRoles Response Body.

Table 661: Keyfactor Command Permissions List

Permission Name	Permission Value
AgentAutoRegistrationModify	Permission to modify agent auto registrations.
AgentAutoRegistrationRead	Permission to read agent auto registrations.
AgentManagementModify	Permission to modify agents.
APIRead	Permission to use the Keyfactor Web APIs.
CertificateCollectionsModify	Permission to modify certificate collections.
CertificateMetadataTypesModify	Permission to modify metadatatypes.
CertificateMetadataTypesRead	Permission to read metadata types.
CertificatesImport	Permission to import certificates.
CertificatesModify	Permission to modify certificates' metadata.
CertificatesRead	Permission to read certificates.
CertificatesRecover	Permission to recover certificates.
CertificatesRevoke	Permission to revoke certificates.
CertificateStoreManagementModify	Permission to modify certificate stores.
CertificateStoreManagementRead	Permission to read certificate stores.
CertificateStoreManagementSchedule	Permission to schedule certificate stores.
MacAutoEnrollManagementModify	Permission to modify Mac auto enrollment settings.
MacAutoEnrollManagementRead	Permission to read Mac auto enrollment settings.
ManagementPortalRead	Permission to read the Keyfactor Command Management Portal.
MonitoringModify	Permission to modify monitoring settings.
MonitoringRead	Permission to read monitoring settings.
MonitoringTest	Permission to test monitoring.
PKIManagementModify	Permission to modify PKI management settings.
PKIManagementRead	Permission to read PKI management settings.
ReportsModify	Permission to modify reports.

Permission Name	Permission Value
ReportsRead	Permission to read reports.
SecuritySettingsModify	Permission to modify security settings.
SecuritySettingsRead	Permission to read security settings.
SSLManagementModify	Permission to modify SSL management settings.
SSLManagementRead	Permission to read SSL management settings.
SystemSettingsModify	Permission to modify system settings.
SystemSettingsRead	Permission to read system settings.
WorkflowModify	Permission to modify alert definitions.
WorkflowParticipate	Permission to approve/deny pending certificates.
WorkflowRead	Permission to read certificates in a pending state and alert definitions.
WorkflowTest	Permission to test alerts.

Table 662: POST /AddRole Request Parameters

Parameter Name	Parameter Value
Name	The name of the security role. This parameter is **required**.
Description	A description of the security role. This parameter is **required**.
Permissions	A list of permissions for the security role. This parameter is optional.
Identities	A list of security identities that will be associated with the security role. This parameter is optional.

Example Request

POST http://<host>/CMSApi/Security/1/AddRole

```
{
       "Name":"<Name>",
       "Description":"<Description>",
       "Permissions": ["<Permission>","<Permission>"],
```

```
        "Identities": ["<Domain>\\<Identity>", "<Domain>\\<Identity>"]
}
```

Example Response

Status Code: 200

```
{
        "Id": <Id>, “Name": "<Name>",
        "Description": "<Description>",
        "Valid": true,"Permissions": "<List of permissions>",
        "Identities": "<List of identities>"
}
```

2.3.7.6 Security EditRole

The POST EditRole endpoint modifies existing security roles. The parameters for the EditRole endpoint can be found in Table 663: POST /EditRole Request Parameters. The administrator role’s name, description and permissions cannot be changed.

Table 663: POST /EditRole Request Parameters

Parameter Name	Parameter Value
Id	The Id of the security role to be edited. This parameter is **required**.
Name	The name to which the security role will be changed. This parameter is optional.
Description	The description of which the security role will be changed. This parameter is optional.
Permissions	The permissions to which the security role will be changed. This parameter is optional.
Identities	The identities to which the security role will be changed. This parameter can take either the Id of a security identity or the identity name. This parameter is optional.

Example Request

POST http://<host>/CMSApi/Security/1/EditRole

```
{
        "Id":<Id>,
```

```
      "Identities":[<List of Identities>]
}
```

Example Response

Status Code: 200

```
{
       "Id": <Id>,
       "Name": "<Name>",
       "Description": "<Description>",
       "Valid": true,
       "Permissions": "<List of Permissions>",
       "Identities": "List of Identities>"
}
```

2.3.7.7 Security DeleteRole

The POST DeleteRole endpoint can be used to delete a security role from Keyfactor Command. A role can be deleted by name or Id. The administrator role cannot be deleted.

Example Request

POST http://<host>/CMSApi/Security/1/DeleteRole

```
{
       "Id": <Id>
}
```

Example Response

Status Code: 200

```
{
       "Message": "Successfully deleted Role: <Name of Role>"
}
```

2.3.8 SSL

Keyfactor Command allows, through the Keyfactor Command Windows Agent, various network segments to be scanned for endpoints serving SSL certificates as well as endpoints presenting a certificate to be monitored for changes in status. An SSL scan is executed against an Endpoint Group, which is a collection of network endpoints, along with a scan schedule. Two types of endpoint groups exist:

- Discovery
  A Discovery endpoint group contains endpoints to be scanned for certificates.
- Monitoring
  A Monitoring group allows endpoints that presented a certificate in a discovery scan to be repeatedly scanned for changes.

The SSL Web API component allows SSL scan configuration to be retrieved and updated in order to facilitate rapid configuration of large numbers of network endpoints. The methods included in this component are given in Table 664: SSL Endpoints. As with the Certstore API component, the SSL component only has 1 version and all endpoints can be accessed through a URL path including */SSL/1/*.

Table 664: SSL Endpoints

Endpoint	Method	Description
AddEndpoint	POST	Add a new endpoint to an endpoint group
AddEndpointGroup	POST	Add a new endpoint group to an agent.
Agents	GET	Return a list of Agents that can perform SSL scans.
EndpointGroups	GET	Returns a list of established endpoint groups for a particular agent

2.3.8.1 SSL AddEndpoint

The AddEndpoing method allows an endpoint to be added to an endpoint group. It returns HTTP 200 OK with response body "true" for successful requests or an appropriate 4xx error with a message on a failure.

Table 665: POST /AddEndpoint Request Body

Parameter Name	Parameter Value
EndpointGroupId	GUID of the endpoint group to which the endpoint should be added, which can be obtained through a combination of the GET SSL/1/Agents and GET SSL/1/EndpointGroups methods.
ItemType	Format in which the network endpoint is defined. Possible values are:

Parameter Name	Parameter Value
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>IPAddress</td></tr><tr><td>2</td><td>DnsName</td></tr><tr><td>3</td><td>NetworkNotation</td></tr></table>
Value	String representing the endpoint. Should be formatted to match the expected format of the ItemType, e.g. "192.168.41.171:443" for IPAddress, "www.example.com:443" for DnsName, or "192.168.0.0/16:443" for NetworkNotation (corresponding to the IP address range 192.168.0.1-192.168.255.254, on port 443 for all endpoints).

Example Request

POST http://<host>/CMSApi/SSL/1/AddEndpoint HTTP/1.1

```
{
       "EndpointGroupId": <GUID>,
       "ItemType": 3,
       "Value": "192.168.0.0/24:443"
}
```

2.3.8.2 SSL AddEndpointGroup

The AddEndpoint Group method allows a new endpoint group to be added for an agent. This requires the two fields shown in Table 666: POST /AddEndpointGroup Request Body. When successful, the GUID and Name of the created endpoint group are returned.

Table 666: POST /AddEndpointGroup Request Body

Parameter Name	Parameter Value
AgentId	GUID of the Agent that will scan endpoints in this group.
FriendlyName	Name of the group to be created.

Table 667: POST /AddEndpointGroup Response Body

Parameter Name	Parameter Value
Guid	Identifier for this endpoint group within Keyfactor Command.
Name	Name of the endpoint group used by Keyfactor Command.

Example Request

POST http://<host>/CMSApi/SSL/1/AddEndpointGroup HTTP/1.1

```
{
       "AgentId": <GUID>,
       "FriendlyName": "local-endpoints"
}
```

Example Response

```
{
       "Guid": "0a44f8af-6808-40ad-9816-d08c2c45d45a",
       "Name": "local-endpoints"
}
```

2.3.8.3 SSL Agents

The Agents HTTP Get method takes no parameters and returns a list of agents that can perform SSL scans. The result will be an array of structures, each with a GUID and name.

Table 668: GET /Agents Response Body

Parameter Name	Parameter Value
Guid	Identifier for this agent within Keyfactor Command.
Name	Hostname of the agent used by Keyfactor Command.

Example Request

GET http://<host>/CMSApi/SSL/1/Agents

Example Response

```
[
	{
		"Guid": "956282ef-f01b-4ae3-8cd2-57327749e15c",
		"Name": "Dev1.jdk.com"
	}
]
```

2.3.8.4 SSL EndpointGroups

The EndpointGroups method returns the list of endpoint groups that have been defined for a particular agent. Unlike most methods in the Keyfactor Web APIs, this is a GET request that takes a parameter as part of the URL query string. The "agentId" required argument is the GUID of the agent for the endpoint groups that should be listed. This value can be retrieved from the GET /CMSApi/SSL/1/Agents response (see SSL Agents on the previous page). The response returned from this method will be an array of endpoint groups with the same structure as the response to AddEndpointGroup (see Table 667: POST /AddEndpointGroup Response Body).

Example Request

GET http://<host>/CMSApi/SSL/1/EndpointGroups?agentId=956282ef-f01b-4ae3-8cd2-57327749e15c HTTP/1.1

Example Response

```
[
	{
		"Guid": "bbf3c3ce-9d7f-48b1-ae5c-c8d38f41d2f1",
		"Name": "MyDiscoveryGroup"
	}
]
```

2.3.9 Workflow

Workflow in Keyfactor Command refers to the process through which pending certificate requests are approved or denied. The Workflow API provides the ability to obtain a list of pending certificate enrollment requests, and approve or deny current requests. This component, like several others, currently encompasses only one version, and methods can all be accessed with the */Workflow/1/* prefix. The methods within this component are listed in Table 669: Workflow Endpoints

Table 669: Workflow Endpoints

Endpoint	Method	Description
Approve	POST	Approve a given pending certificate request

Endpoint	Method	Description
Deny	POST	Deny a given pending certificate request
PendingList	POST	Retrieve a list of outstanding pending certificate requests
Status	GET	Synonym for GET CMSApi/Status (see Status on page 1479)

2.3.9.1 Workflow Approve and Deny

The Approve POST method will attempt to approve the provided pending certificate enrollment request(s), while POST Deny will attempt to deny the request(s). In both cases, the structure of the pending request(s) is the same—an array of pending certificate enrollment requests must be provided in the format given in Table 671: POST /Approve and /Deny PendingRequests Details. If only one request is to be sent, it should be provided as a list with one element. The one difference between the request formats for the two methods is that Deny supports an optional "Comments" field, which provides an opportunity to describe the reason for the request denial, shown in Table 670: POST /Approve and /Deny Request Body. In both cases, an array of successful, failed and forbidden requests will be returned. The method will accept various inputs used to qualify the request to be approved, as shown in Table 671: POST /Approve and /Deny PendingRequests Details.

Table 670: POST /Approve and /Deny Request Body

Parameter Name	Parameter Value
PendingRequests	Array of requests to be approved or denied. Required for both methods.
Comments	String describing the reason for the request denial. Optional for Deny and not permitted for Approve.

Table 671: POST /Approve and /Deny PendingRequests Details

Parameter Name	Parameter Value
CMSRequestId	The Keyfactor Command request database identifier. This parameter is the most specific, and can be used without any other parameters provided. An exception will be returned if this identifier is not found within Keyfactor Command.
CAHost	Host name of the certificate authority against which the certificate enrollment request was submitted. This parameter also requires the CALogicalName and CARequestId parameters to be provided in the request. An exception will be returned if a certificate authority with this host, logical name and request ID is not found within Keyfactor Command.
CALogicalName	Logical name of the certificate authority against which the certificate enrollment request was submitted. This parameter also requires the CAHost and CARequestId parameters to be provided in the request. An exception will be returned if a certificate authority with this host, logical name and

Parameter Name	Parameter Value
	request ID is not found within Keyfactor Command.
CARequestId	Request/row identifier of the request for certificate authority defined by CAHost and CALogicalName. This parameter also requires the CALogicalName and CAHost parameters to be provided in the request. An exception will be returned if a certificate authority with this host, logical name and request ID is not found within Keyfactor Command.

Table 672: POST /Approve and /Deny Response Body

Parameter Name	Parameter Value
Successes	An array of the successful approval response details (see table below in this section).
Failures	An array of the failed approval response details (see table below in this section). Failures of this type are generally exceptions.
Denials	An array of the approval requests that were denied (see table below in this section). Denials are usually created by insufficient user permissions required to perform the approval.

Table 673: POST /Approve and /Deny Result Details

Parameter Name	Parameter Value
CAHost	Host name of the certificate authority against which the certificate enrollment request was submitted.
CALogicalName	Logical name of the certificate authority against which the certificate enrollment request was submitted.
CMSRequestId	The Keyfactor Command request database identifier.
CARequestId	Request/row identifier of the request for certificate authority defined by CAHost and CALogicalName.
Comment	Brief description of the reason for the failure or denial, or simply ‘Success’ if the request succeeded.

Example Request

Providing only a CMSRequestId

POST http://<host>/CMSApi/Workflow/1/Approve HTTP/1.1

```
{
       "PendingRequests":
       [
       {"CMSRequestId": <cms-request-id1>},
       {"CMSRequestId": <cms-request-id2>}
       ]
}
```

Example Request

Providing the certificate authority information

POST http://<host>/CMSApi/Workflow/1/Approve HTTP/1.1

```
{
       "PendingRequests":
       [{
       "CAHost":"<ca-host>","CALogicalName":"<ca-name>","CARequestId":<ca-request-id>
       }]
}
```

Example Request

Providing both types of information

POST http://<host>/CMSApi/Workflow/1/Approve HTTP/1.1

```
{
       "PendingRequests":
       [
       {"CMSRequestId":<cms-request-id1>},
       {"CAHost":"<ca-host>","CALogicalName":"<ca-name>","CARequestId":<ca-request-id>}
       ]
}
```

Example Response

(Successful)

```
{
       "Successes":
       [{
```

```
        "CAHost": "<ca-host>",
        "CALogicalName": "<ca-name>",
        "CARequestId": <ca-request-id>,
        "Comment": "Successful"
        }],
        "Failures": [],
        "Denials": []
}
```

Example Response

(Invalid identifier)

```
{
        "Successes": [],
        "Failures": [{
        "CAHost": "<ca-host>",
        "CALogicalName": "<ca-name>",
        "CARequestId": <ca-request-id>,
        "CMSRequestId": <cms-request-id>,
        "Comment": "Unable to approve the request: <ca request id> for the certificate authority: '<ca-host-
name>\<ca-logical-name>' \r\nfor the current user: '<requester>': No request for: CMS Request Id: 0,
CA Host: <ca-host>, CA Logical Name: <ca-name>, CA Request Id: <ca-request-id>"
        }],
        "Denials": []
}
```

2.3.9.2 PendingList

The POST PendingList method will return the current set of pending certificate enrollment requests stored within Keyfactor Command matching the provided parameters. The response will be a JSON object with a single field , PendingRequests, mapped to an array where each entry represents a single pending certificate request that matches the parameters provided in the HTTP request. Each of these entries will have the format given in Table 675: POST /PendingList Response Body.

Table 674: POST /PendingList Request Body

Parameter Name	Parameter Value
CAHost	Host name of the certificate authority against which the certificate enrollment request was submitted. This parameter also requires the CALogicalName parameter to be provided in the request. An exception will be returned if a certificate authority with this host and logical name is not found within Keyfactor Command.

Parameter Name	Parameter Value
CALogicalName	Logical name of the certificate authority against which the certificate enrollment request was submitted. This parameter also requires the CAHost parameter to be provided in the request. An exception will be returned if a certificate authority with this host and logical name is not found within Keyfactor Command.
LowerDate	Any pending requests prior to this date should be ignored. Optional.
UpperDate	Any pending requests after this date should be ignored. Optional.

Table 675: POST /PendingList Response Body

Parameter Name	Parameter Value
CAHost	Host name of the certificate authority against which the certificate enrollment request was submitted.
CALogicalName	Logical name of the certificate authority against which the certificate enrollment request was submitted.
CARequestId	Identifier associated with the request within the certificate authority.
CertificateAuthority	Combination of the CAHost and CALogicalName (CAHost\\CALogicalName).
CMSRequestId	Database identifier associated with the request within Keyfactor Command.
CommonName	Common name requested for the certificate.
DistinguishedName	Distinguished name requested for the certificate.
TemplateName	Certificate template for which the certificate was requested.
KeySize	Number of bits in the certificate's private key.
Requester	User or principal who requested the certificate, generally formatted "DOMAIN\\user".
SubmissionDate	ISO-8601 formatted timestamp at which the certificate request was received.
SubjectAlternativeName	Array of SANs requested for the certificate. The entries each correspond to one requested SAN element, and each one will be in the form given in Table 676: POST /PendingList SubjectAlternativeName Details

Table 676: POST /PendingList SubjectAlternativeName Details

<table>
<tr><th>Parameter Name</th><th>Parameter Value</th></tr>
<tr><td>Type</td><td>Type of this SAN element on the certificate request. Will take one of the following values:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Other</td></tr>
<tr><td>1</td><td>RFC 822 name (e-mail address)</td></tr>
<tr><td>2</td><td>DNS name</td></tr>
<tr><td>3</td><td>X400 address</td></tr>
<tr><td>4</td><td>Directory Name</td></tr>
<tr><td>5</td><td>Edi Party Name</td></tr>
<tr><td>6</td><td>URI</td></tr>
<tr><td>7</td><td>IP address</td></tr>
<tr><td>8</td><td>Registered ID</td></tr>
<tr><td>100</td><td>Microsoft NT Principal Name</td></tr>
<tr><td>101</td><td>Microsoft NTDS Replication</td></tr>
<tr><td>999</td><td>Unknown</td></tr>
</table></td></tr>
<tr><td>Value</td><td>String representation of the value requested for this SAN element.</td></tr>
</table>

Example Request

POST http://<host>/CMSApi/Workflow/1/PendingList HTTP/1.1

```
{
       "CAHost": "<ca-host>",
       "CALogicalName": "<ca-name>",
       "LowerDate": <date or null or left out completely>,
       "UpperDate": <date or null or left out completely>
}
```

Example Response

```
{
        "PendingRequests":
        [{
        "CAHost":"<ca-host>",
        "CALogicalName":"<ca-name>",
        "CARequestId":<ca-request-id>,
        "CMSRequestId":<cms-request-id>
        }]
}
```

2.3.10 Workflow Expiration Alerts

Workflow in Keyfactor Command refers to the process through which pending certificate requests are approved or denied. The Workflow Expiration Alert APIs provides the ability to manage expiration alerts, event handlers, registered event handlers and schedules.

2.3.10.1 Workflow Expiration Alerts Endpoints

The Workflow Expiration Alert API provides the ability to list, create, update and delete expiration alerts for Keyfactor Command via the Keyfactor API. The methods within this component are listed in Table 677: Workflow Expiration Alerts Endpoints

Table 677: Workflow Expiration Alerts Endpoints

Endpoint	Method	Description
ExpirationAlerts	GET	List all Expiration Alerts or a get a single expiration alert definition.
ExpirationAlerts	POST	Create a new expiration alert definition.
ExpirationAlerts	PUT	Update an existing expiration alert definition.
ExpirationAlerts	DELETE	Delete an existing expiration alert definition.

Workflow Expiration Alerts

Note: For the GET (single), PUT, and DELETE methods you will need the expiration alert ID. You will need to run the GET (list) method to acquire the ID in order to proceed with those methods.

Note: For the POST and PUT methods, if you are using Registered Event Handlers, you will need to run the Event handler GET (list) method to acquire the ID prior to issuing the expiration alert method (see Workflow Expiration Alert Handler Parameters Endpoints on page 1474).

Table 678: Workflow Expiration Alert Parameters

Parameter Name	Parameter Value
Id/Alert ID	The database ID of the Alert
DisplayName	Alert display name
Subject	The subject field of the alert
Message	The message field of the alert
UseHandler	True/False, whether or not the Use Handler checkbox is checked for the alert
Days	The number of days to alert before expiration
RegisteredEventHandlerId	Id of the Event Handler to use. See (Workflow Expiration Alert Handler Parameters Endpoints on page 1474)
CertificateQuery	Name, and/or Id, of the certificate collection of the alert
ExpirationAlertRecipients	Id and/or Recipient email address in a comma separated list of objects. So there could be multiple addresses chunks in curly brackets{}, comma separated in the array in the square brackets []

LIST all expiration alerts:

Example Request

GET ~/ExpirationAlerts/1/List?page=<page number>&returnlimit=<max results to get>&sortname=<field to sort by>&sortorder=<asc or desc>

```
no body
```

Example Response

```
[
  {
    "Id": <id>,
    "DisplayName": "Alert display name",
    "QueryName": "Certificate query name",
    "Days": <number of days to alert before expiration>,
    "HandlerName":  "Name of the Event Handler if any"
```

```
  }
]
```

Get a single alert definition

Example Request

GET ~/ExpirationAlerts/1/<Alert Id>

```
no body
```

Example Response

```
[
  {
    "Id": <id>,
    "DisplayName": "Alert display name",
    "Subject": "Alert Subject",
    "Message": "Alert message body",
    "UseHandler":  <true/false>,
    "Days": <number of days to alert before expiration>,
    "RegisteredEventHandlerId":  <Id of the Event Handler to use>,
    "CertificateQuery": { "Id": <Cert query id>, "Name": "Cert query name" },
    "ExpirationAlertRecipients": [ { "Id": <recipient Id>, "Email": "Recipient email address" } ]
  }
]
```

Create New Expiration Alert

Example Request

POST ~/ExpirationAlerts/1/

```
 {
    "DisplayName": "Alert display name",
    "Subject": "Alert Subject",
```

```
  "Message": "Alert message body",
  "UseHandler":  <true/false>,
  "Days": <number of days to alert before expiration>,
  "RegisteredEventHandlerId":  <Id of the Event Handler to use>,
  "CertificateQuery": { "Id": <Cert query id> },
  "ExpirationAlertRecipients": [ { "Email": "Recipient email address" } ]
}
```

Example Response

```
{
  "Id": <id>,
  "DisplayName": "Alert display name",
  "Subject": "Alert Subject",
  "Message": "Alert message body",
  "UseHandler":  <true/false>,
  "Days": <number of days to alert before expiration>,
  "RegisteredEventHandlerId":  <Id of the Event Handler to use>,
  "CertificateQuery": { "Id": <Cert query id>, "Name": "Cert query name" },
  "ExpirationAlertRecipients": [ { "Id": <recipient Id>, "Email": "Recipient email address" } ]
}
```

Update Existing Expiration Alert

Example Request

PUT ~/ExpirationAlerts/1/<Alert Id>

```
{
   "DisplayName": "Alert display name",
   "Subject": "Alert Subject",
   "Message": "Alert message body",
   "UseHandler":  <true/false>,
   "Days": <number of days to alert before expiration>,
   "RegisteredEventHandlerId":  <Id of the Event Handler to use>,
   "CertificateQuery": { "Id": <Cert query id> },
   "ExpirationAlertRecipients": [ { "Email": "Recipient email address" } ]
 }
```

Example Response

```
{
  "Id": <id>,
  "DisplayName": "Alert display name",
  "Subject": "Alert Subject",
  "Message": "Alert message body",
  "UseHandler":  <true/false>,
  "Days": <number of days to alert before expiration>,
  "RegisteredEventHandlerId":  <Id of the Event Handler to use>,
  "CertificateQuery": { "Id": <Cert query id>, "Name": "Cert query name" },
  "ExpirationAlertRecipients": [ { "Id": <recipient Id>, "Email": "Recipient email address" } ]
}
```

Delete Expiration Alert

Example Request

DELETE ~/ExpirationAlerts/1/<Alert Id>

```
no body
```

Example Response

204 No Content

2.3.10.2 Workflow Expiration Alert Event Handler Parameters API

The Workflow Expiration Alert Event Handler Parameter API provides the ability to list, create, update and delete expiration alert event handler parameters for specific Keyfactor Command expiration alerts via the Keyfactor API. The methods within this component are listed in Table 679: Workflow Expiration Alerts Event Handler Parameters Endpoints

Table 679: Workflow Expiration Alerts Event Handler Parameters Endpoints

Endpoint	Method	Description
HandlerParameters	GET	List all, or a given, expiration alert Handler Parameter(s) for an expiration alert.
HandlerParameters	POST	Create a new handler parameter for an expiration alert.
HandlerParameters	PUT	Update an existing expiration alert handler parameter for an expiration alert.
HandlerParameters	DELETE	Delete an existing expiration alert handler parameter for an expiration alert.

Workflow Expiration Alert Handler Parameters Endpoints

Note: For the GET (single), PUT, and DELETE methods you will need the handler parameter ID. You will need to run the GET (list) method to acquire the ID in order to proceed with those methods.

Table 680: Workflow Expiration Alert Handler Parameters

Parameter Name	Parameter Value
Id/Alert ID	The database ID of the handler parameter
Key	The parameter name
DefaultValue	The given value for the handler parameter
ParameterType	The event handler parameter type number Special Text = 0 Static Value = 1 PowerShell Script Name = 2 Logging Target Machine = 3 Renewal URL = 4 Renewal Template = 5 Renewal Certificate Authority = 6
ExpirationAlertDefinitionId	The database ID of the expiration alert definition

List All Handler Parameters for an Expiration Alert:

Example Request

GET ~/ExpirationAlerts/1/<Alert Id>/HandlerParameters/List?page=<page number>&returnlimit=<max results to get>&sortname=<field to sort by>&sortorder=<asc or desc>

```
no  body
```

Example Response

```
[
  {
    "Id": <id>,
    "Key": "Parameter Key name",
    "DefaultValue": "default value for parameter",
    "ParameterType": <Event Handler Parameter Type number>,
    "ExpirationAlertDefinitionId":  <alert Id>
  }
]
```

Get Handler Parameter by Id

Example Request

GET ~/ExpirationAlerts/1/<Alert Id>/HandlerParameters/<handler param Id>

```
no body
```

Example Response

```
[
  {
    "Id": <id>,
    "Key": "Parameter Key name",
    "DefaultValue": "default value for parameter",
    "ParameterType": <Event Handler Parameter Type number>,
    "ExpirationAlertDefinitionId":  <alert Id>
  }
]
```

Create New Handler Parameter

Example Request

POST ~/ExpirationAlerts/1/<Alert Id>/HandlerParameters

```
  {
    "Key": "Parameter Key name",
    "DefaultValue": "default value for parameter",
    "ParameterType": <Event Handler Parameter Type number>,
   }
```

Example Response

```
{
    "Id": <id>,
    "Key": "Parameter Key name",
    "DefaultValue": "default value for parameter",
    "ParameterType": <Event Handler Parameter Type number>,
   }
```

Update Existing Handler Parameter

Example Request

PUT ~/ExpirationAlerts/1/<Alert Id>/HandlerParameters/<handler param Id>

```
  {
    "Key": "Parameter Key name",
    "DefaultValue": "default value for parameter",
    "ParameterType": <Event Handler Parameter Type number>,
   }
```

Example Response

```
{
    "Id": <id>,
    "Key": "Parameter Key name",
    "DefaultValue": "default value for parameter",
    "ParameterType": <Event Handler Parameter Type number>,
   }
```

Delete Handler Parameter

Example Request

DELETE ~/ExpirationAlerts/1/<Alert Id>/HandlerParameters/<handler param Id>

```
no  body
```

Example Response

204 No Content

2.3.10.3 Workflow Expiration Alert Registered Event Handlers API

The Workflow Expiration Alert Registered Event Handlers API provides the ability to list expiration alert registered event handlers: for Keyfactor Command via the Keyfactor API. The methods within this component are listed in Table 681: Workflow Expiration Alerts Registered Event Handlers Endpoints

Table 681: Workflow Expiration Alerts Registered Event Handlers Endpoints

Endpoint	Method	Description
RegisteredEventHandlers	GET	Get list of Registered Event Handlers

Workflow Expiration Alert Registered Event Handlers Parameters

Table 682: Workflow Expiration Alert Registered Event Handlers Parameters

Parameter Name	Parameter Value
Id	The database ID of the registered event handler
Classname	Fully qualified class name of the registered event handler implementation in the associated assembly
DisplayName	The display name of registered event handler
Enabled	True/False, whether or not the Use Handler checkbox is checked for the alert
RegisteredEventAssemblyId	The Id of the registered event handler assembly

LIST all Registered Event Handlers:

Example Request

GET ~/ExpirationAlerts/1/RegisteredEventHandlers/List?page=<page number>&returnlimit=<max results to get>

```
no body
```

Example Response

```
{
  "Id": <registered event handler Id>,
  "ClassName": "class name of event handler",
  "DisplayName": "display name of event handler",
  "Enabled": <true/false>,
  "RegisteredEventAssemblyId":  <id of the registered event assembly>
}
```

2.3.10.4 Workflow Expiration Alert Schedule API

The Workflow Expiration Alert Schedule API provides the ability to list, create, and set expiration alert schedules for Keyfactor Command via the Keyfactor API. The methods within this component are listed in Table 683: Workflow Expiration Alerts Schedule Endpoints

Table 683: Workflow Expiration Alerts Schedule Endpoints

Endpoint	Method	Description
Schedule	GET	Get the schedule set for all expiration alerts.
Schedule	POST	Create a new schedule for an expiration alert.

Workflow Expiration Alert Schedule Parameters

Table 684: Workflow Expiration Alert Schedule Parameters

Parameter Name	Parameter Value
Daily	The display name of registered event handler
Time	The ISO string of time to schedule run

LIST Expiration Alert Schedule

Example Request

GET ~/ExpirationAlerts/1/Schedule

```
no body
```

Example Response

```
{
  "Daily": {
    "Time":"ISO string of time to schedule run"
   }
}
```

Set Alerts Schedule

Example Request

POST ~/ExpirationAlerts/1/Schedule

```
{
  "Daily": {
    "Time":"ISO string of time to schedule run"
   }
}
```

Example Response

204 No Content

2.3.11 Status

The Status Web API component provides a single method to retrieve various aspects of the Keyfactor Command server state. This method is an HTTP GET Status request with no parameters required. As of Keyfactor Command 5.0, the Status endpoint generally is not needed by a Web API client application, as the Keyfactor Command version is passed back in an HTTP header with every response to every Web API request. However, it is included to preserve compatibility with applications already using it or applications requiring more information.

Example Request

GET http://<host>/CMSApi/Status HTTP/1.1

Example Response

Status Code: 200


```
{
	"ApiMajorRev": 2,
	"ApiMinorRev": 0,
	"ProductMajorVersion": 5,
	"ProductMinorVersion": 0,
	"ProductBranchVersion": 0,
	"ProductBuildVersion": 1,
	"LicenseStatus": "Licensed",
	"Modules": [{
		"Name": "CertEnroll",
		"Versions": [1, 2, 3]
	},
	{
		"Name": "Certificates",
		"Versions": [1, 2, 3]
	},
	{
		"Name": "CertStore",
		"Versions": [1]
	},
	{
		"Name": "Metadata",
		"Versions": [2, 3]
	},
	{
		"Name": "Ssl",
		"Versions": [1]
	},
	{
		"Name": "Status",
		"Versions": null
	},
	{
		"Name": "Workflow",
		"Versions": [1]
	}]
}
```

2.3.12 vSCEP

The vSCEP API method supports enrollment through the Keyfactor Command implementation of the SCEP protocol. The single method—GET CMSValidation/api/vSCEP—is used to retrieve a SCEP challenge, while also associating that challenge with the specified certificate subject information. This method differs from the other Web API methods in that it is not included in the CMSApi virtual directory, but in the separate "CMSValidation/api" directory. It also differs in that, while it is a GET method, it does take request parameters, which means that these parameters must be URL-encoded in the query string. Like the other Web API methods, however, it requires the Accept and Authorization headers, and returns a 200 OK status if a connection was successfully made to the vSCEP server or an appropriate 4XX status if a connection could not be made. The request and response formats are given in the below tables and example. All fields in the request are optional, and all but the Subject parameter may be submitted multiple times (for example, to include two different DNS SANs in the same certificate).

Table 685: GET /CMSValidation/api/vSCEP Query String Parameters

Parameter Name	Parameter Value
Subject	Distinguished Name that should be used as the certificate subject.
DNS	Subject Alternative Name representing a DNS record.
IP	Subject Alternative Name representing an IP address.
RFC822	Subject Alternative Name representing an RFC822 Name (email address).
NTPrincipal	Subject Alternative Name representing an NT Principal Name.

Table 686: GET /CMSValidation/api/vSCEP Response Body

Parameter Name	Parameter Value
Status Code	HTTP Status Code vSCEP received from the SCEP server. This will be 200 if the request was successful.
Message	Status message for the request. In the case of an error retrieving a SCEP challenge, this will provide more detailed error information.
Challenge	SCEP Challenge represented as a hex string. In the case of an error, this will be null.
Hash	MD5 hash of the CA certificate associated with the SCEP server. In the case of an error, this will be null.

Example Request

GET http://<host>/CMSValidation/api/vSCEP?subject=CN%3DBob%20Smith%2CO%3DExample%20Company&RFC822=bob.smith%40mail.example.com HTTP/1.1

Example Response

Status Code: 200

```
{
       "Challenge":"247FAFEEABA1F9B7",
       "Hash":"01940B86 9C6C03DC 79BF2E5B 741779DF",
       "StatusCode":200,
       "Message":"Request stored successfully"
}
```

2.4 API Change Log

In this section you will find the change history for the Keyfactor Command API endpoints from version 9.0 on.

Find the change log for Keyfactor API below.

2.4.1 v9 API Change Log

Find the version 9 change log for Keyfactor API below.

Link to Change Logs
API Change Log v9.0 below
API Change Log v9.1 on page 1484
API Change Log v9.2 on page 1485
API Change Log v9.3 on page 1485
API Change Log v9.4 on page 1486
API Change Log v9.5 on page 1486
API Change Log v9.6 on page 1486
API Change Log v9.7 on page 1486
API Change Log v9.8 on page 1486
API Change Log v9.9 on page 1486

2.4.1.1 API Change Log v9.0

API changes for Keyfactor Command version 9.0 Major release

Table 687: 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	
/Reports/{id}	GET	Add	

Endpoint	Method	Action	Notes
/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	

2.4.1.2 API Change Log v9.1

API changes for Keyfactor Command version 9.1 incremental release

Table 688: 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	

2.4.1.3 API Change Log v9.2

API changes for Keyfactor Command version 9.2 incremental release

Table 689: 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.

2.4.1.4 API Change Log v9.3

API changes for Keyfactor Command version 9.3 incremental release

Table 690: API Change Log v9.3

Endpoint	Methods	Action	Notes
/JobTypes/Custom	POST	Fix	No longer requires default field values.

2.4.1.5 API Change Log v9.4

API changes for Keyfactor Command version 9.4 incremental release

Table 691: API Change Log v9.4

Endpoint	Methods	Action	Notes
/Workflow/Certificates/Pending	GET	Update	Now returns the associated metadata.

2.4.1.6 API Change Log v9.5

API changes for Keyfactor Command version 9.5 incremental release

Table 692: API Change Log v9.5

Endpoint	Methods	Action	Notes
/Enrollment/PFX	POST	Update	No longer requires a certificate authority name to be provided.

2.4.1.7 API Change Log v9.6

API changes for Keyfactor Command version 9.6 incremental release.

No API endpoint changes were made in this release.

2.4.1.8 API Change Log v9.7

API changes for Keyfactor Command version 9.7 incremental release

Table 693: API Change Log v9.7

Endpoint	Methods	Action	Notes
/KeyfactorAPI/License	GET	Add	

2.4.1.9 API Change Log v9.8

API changes for Keyfactor Command version 9.8 incremental release.

No API endpoint changes were made in this release.

2.4.1.10 API Change Log v9.9

API changes for Keyfactor Command version 9.9 incremental release

Table 694: 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.

2.4.2 v10 API Change Log

Find the version 10 change log for Keyfactor API below.

Link to Change Logs
API Change Log v10.0 below

2.4.2.1 API Change Log v10.0

API changes for Keyfactor Command version 10.0 Major release

Table 695: 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	
/Alerts/KeyRotation/{id}	GET, DELETE	Add	
/Alerts/KeyRotation/Schedule	GET, PUT	Add	

Endpoint	Methods	Action	Notes
/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	
/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

Endpoint	Methods	Action	Notes
			the /CertificateStores API endpoints directly as JobProperties using the Properties parameter, replacing the deprecated /CertificateStores/Server API endpoints.
/CertificateStores	GET, POST, PUT	Updated	Server usernames, server passwords, and the UseSSL flag are managed by the /CertificateStores API endpoints directly as JobProperties using the Properties parameter, replacing the deprecated /CertificateStores/Server API endpoints.
/Enrollment/PFX (v2)	POST	Add	
/Enrollment/Settings/{id}	GET	Add	
/JobTypes/Custom	POST	Update	DefaultValue property is no longer required, validation is now performed on the JobTypeFields/DefaultValue property, validation prevents names containing spaces.
/JobTypes/Custom/{id}	DELETE	Update	Includes validation so that deletion is prevented if at least one associated approved orchestrator implements the capability.
/MacEnrollment	GET, PUT	Add	
/Monitoring/Revocation	GET, POST	Update	Renamed from /Workflow/RevocationMonitoring
/Monitoring/Revocation/{id}	GET, PUT, DELETE	Update	Renamed from /Workflow/RevocationMonitoring/{id}
/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	

Endpoint	Methods	Action	Notes
/OrchestratorJobs/Unschedule	POST	Add	
/OrchestratorJobs/Acknowledge	POST	Add	
/Security/Identities/{id}	GET	Add	
/Security/Roles/{id}/Identities	GET, POST	Add	
/Security/Roles/{id}/Containers	GET, POST	Add	
/Security/Roles/{id}/Copy	POST	Add	
/Security/Roles/{id}/Permissions	GET	Add	
/Security/Roles/{id}/Permissions/Global	GET, POST, PUT	Add	
/Security/Roles/{id}/Permissions/Collections	GET, POST, PUT	Add	Replaced the /CertificateCollections/{id}/Permissions endpoint functionality.
/Security/Roles/{id}/Permissions/Containers	GET, POST, PUT	Add	Returns only containers that have a permission set for the selected security role.
/SMTP	GET, PUT	Add	
/SMTP/Test	POST	Add	
/Templates	GET, PUT	Update	Includes template-specific policy information.
/Templates/{id}	GET	Update	Includes template defaults.
/Templates/Settings	GET, PUT	Update	Includes global template policies.
/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	

Endpoint	Methods	Action	Notes
/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	

2.4.2.2 API Change Log v10.1

API changes for Keyfactor Command version 10.1 incremental release

Table 696: 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.

2.4.2.3 API Change Log v10.2

API changes for Keyfactor Command version 10.2 incremental release

Table 697: API Change Log v10.2

Endpoint	Methods	Action	Notes
/Security/My	GET	Add	Returns all the security roles and global permissions for the requesting user.
/Enrollment/CSR	POST	Update	The workflow instance ID has been added to the response.
/Enrollment/CSR	POST	Update	A new PrivateKey input field has been added to support private key retention on CSR enrollment.
/Enrollment/PFX	POST	Update	The workflow instance ID has been added to the response.
/Certificates/Analyze	POST	Update	The endpoint requires Global Certificates-Read or Certificates-Import permissions.

3.0 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. server-name.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. server-name.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. server-name.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 Windows servers (a.k.a. IIS certificate stores) and FTP capable 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 run custom jobs to provide certificate management capabilities on a variety of platforms and devices (e.g. F5 devices, NetScaler devices, Amazon Web Services (AWS) resources) and execute tasks outside the standard list of certificate management functions. It runs on either Windows or Linux.

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