Oracle® Key Vault

RESTful Services Administrator's Guide

Release 21.14
G49987-01
April 2026

Oracle Key Vault RESTful Services Administrator's Guide, Release 21.14

G49987-01

Primary Author: Puneet Rai

Contributors: Sunil Surabhi, Prakash Jashnani, Mark Doran, Patricia Huey

Contributors: Alexis Abell, Bharathi Baskaran, Shubham Goyal, Rahil Mir, Dongwon Park, Vipin Samar, Radhika Siravara, Ajay Srivastava, Venkatesh Petla, Peter Wahl, Daniel Wu, Abhishek A

Contents

3 Administration Commands

4 Access Management Commands

5 Security Object Commands

6 Cryptographic Commands

7 Monitoring Commands

8 Cluster Monitoring Commands

9 Cluster Management Commands

10 Backup, Schedule, and Restore Commands

Preface

Welcome to *Oracle Key Vault RESTful Services Administrator's Guide* . This guide explains how to use the Oracle Key Vault RESTful services to manage endpoints, wallets, security objects, deployment, and Oracle Key Vault general tasks such as performing backup operations.

Audience

Oracle Key Vault is meant for users who are responsible for deploying, maintaining, and managing security within the enterprise. These users can be database, system, or security administrators. This guide can be used by any information security personnel who is responsible for protecting enterprise data residing in database servers, application servers, operating systems, and other information systems.

Related Documents

For more information, see these Oracle resources:

- *Oracle Key Vault Administrator's Guide*
- *Oracle Key Vault Root of Trust HSM Configuration Guide*
- *Oracle Key Vault Developer's Guide*
- *Oracle Key Vault Licensing Information*
- *Oracle Key Vault Release Notes*
- *Key Management Interoperability Protocol Specification Version 1.1*

To download the product data sheet, frequently asked questions, links to the latest product documentation, product download, and other collateral, visit Oracle Technical Resources (formerly Oracle Technology Network). You must register online before using Oracle Technical Services. Registration is free and can be done at

`https://www.oracle.com/technical-resources/`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release for Oracle Key Vault

This Oracle Key Vault release introduces new features that enhance the use of Oracle Key Vault in a large enterprise.

Changes for Oracle Key Vault Release 21.13

Oracle Key Vault release 21.13 introduces several new features.

Renamed Endpoint Installation Options for Oracle AI Database 26ai New Feature Support

Oracle AI Database 26ai uses the OpenSSL FIPS module when operating in FIPS mode for cryptographic operations.

Starting with release 21.11, Oracle Key Vault provides an OpenSSL-based PKCS#11 library that you can use in FIPS mode with Oracle AI Database 26ai. Additionally, Oracle Key Vault

supports the newer version of the local auto-login wallet introduced with Oracle AI Database 26ai.

Starting with Oracle Key Vault release 21.13, the installation options for endpoint software to support newer features of Oracle AI Database 26ai are renamed as follows:

- When installing using the downloaded `okvclient.jar`, use the `-arch db26ai` option instead of `-arch db23ai`.
- When installing using the RESTful Services utility command `okv admin endpoint provision`, use the `--arch db26ai` option instead of `--arch db23ai`.

Oracle Key Vault continues to support the previous `db23ai` option for backward compatibility, but recommends that you use `db26ai`.

Changes for Oracle Key Vault Release 21.11

Oracle Key Vault release 21.11 introduces several new features.

- Protection for Security Objects During Register and Download Operations
  Starting with release 21.11, you can protect security objects when registering or downloading them from Oracle Key Vault. Objects are protected using a symmetric key stored in Oracle Key Vault.
- Searching for Security Objects With Their Date Attributes Within a Specified Date Range
  Starting with release 21.11, you can search for security objects based on their date attributes, such as activation date, deactivation date, etc., that fall within a specified range.
- Support for Cryptographic Parameters Attribute
  Starting with release 21.11, the Oracle Key Vault RESTful services utility provides a new option `--crypto-parameter` that you can use to specify cryptographic parameter attributes with several managed-object commands such as `register`, `get`, `locate`, and `fetch`.
- Listing Status of Each Node in a Cluster with One Command
  Starting with release 21.11, the Oracle Key Vault RESTful services utility provides a new option to list the status of each cluster node with one command.
- System Time of Oracle Key Vault Server
  Starting with release 21.11, the output of the `okv server info get` command always includes the `serverTime` - the current system time of Oracle Key Vault server.

Protection for Security Objects During Register and Download Operations

Starting with release 21.11, you can protect security objects when registering or downloading them from Oracle Key Vault. Objects are protected using a symmetric key stored in Oracle Key Vault.

During object registration, you provide an encrypted object and an unwrapping key. Oracle Key Vault uses the unwrapping key to decrypt the object before registering.

Following commands are updated to include the new option `--unwrap-key-uuid`:

- okv managed-object certificate register
- okv managed-object certificate-request register
- okv managed-object key register
- okv managed-object opaque register
- okv managed-object private-key register

- okv managed-object public-key register
- okv managed-object secret register

During object download, you provide a wrapping key. Oracle Key Vault uses the wrapping key to encrypt the object before returning.

Following commands are updated to include the new option `--wrap-key-uuid`

- okv managed-object certificate get
- okv managed-object certificate-request get
- okv managed-object key get
- okv managed-object opaque get
- okv managed-object private-key get
- okv managed-object public-key get
- okv managed-object secret get

Specify the UUID of the symmetric key used for wrapping or unwrapping with the new options.

Searching for Security Objects With Their Date Attributes Within a Specified Date Range

Starting with release 21.11, you can search for security objects based on their date attributes, such as activation date, deactivation date, etc., that fall within a specified range.

When you provide a date range for a date attribute, the result includes all objects whose corresponding date attribute value falls within the specified range, including the start and end dates.

You can specify date range for date attributes for below commands:

- okv managed-object object fetch
- okv managed-object object locate

Support for Cryptographic Parameters Attribute

Starting with release 21.11, the Oracle Key Vault RESTful services utility provides a new option `--crypto-parameter` that you can use to specify cryptographic parameter attributes with several managed-object commands such as `register`, `get`, `locate`, and `fetch`.

Following commands are enhanced to support the new option:

- `okv managed-object <object-type> register`
- `okv managed-object <object-type> get`
- `okv managed-object object locate`
- `okv managed-object object fetch`

Listing Status of Each Node in a Cluster with One Command

Starting with release 21.11, the Oracle Key Vault RESTful services utility provides a new option to list the status of each cluster node with one command.

You specify the new option `--all-nodes` to the `okv cluster service monitor` command to get the status of all cluster nodes. Earlier, this command allowed you to obtain the status of only one cluster node at time using the `--node-name` option.

System Time of Oracle Key Vault Server

Starting with release 21.11, the output of the `okv server info get` command always includes the `serverTime` - the current system time of Oracle Key Vault server.

In earlier releases, serve time was included only when the command was run by a user with the System Administrator role.

Changes for Oracle Key Vault Release 21.10

Oracle Key Vault release 21.10 introduces several new features.

- New Oracle Key Vault Endpoint Platforms - Linux for Arm (aarch64) Architecture and IBM: Linux on System z
  Starting with Oracle Key Vault release 21.10, you can provision Oracle Key Vault endpoints on the Linux for Arm (aarch64) architecture and IBM: Linux on System z.
- List Recently Active or Inactive Endpoint using the RESTful Services Utility Command
  Starting with Oracle Key Vault release 21.10, you can list the endpoints that were active or inactive recently using the RESTful services utility.

New Oracle Key Vault Endpoint Platforms - Linux for Arm (aarch64) Architecture and IBM: Linux on System z

Starting with Oracle Key Vault release 21.10, you can provision Oracle Key Vault endpoints on the Linux for Arm (aarch64) architecture and IBM: Linux on System z.

Oracle Databases running on Linux for Arm (aarch64) or IBM: Linux on System z can also be enrolled as Oracle Key Vault endpoints for TDE key management. You can set up endpoints for SSH servers or clients on both these platforms. All the endpoint software like `okvutil`, PKCS#11 library, the RESTful services utility, Java, and C SDK are available on these platforms.

The new endpoint platforms are supported with the Oracle Key Vault 21 RESTful services utility but not available with the deprecated classic RESTful services utility.

List Recently Active or Inactive Endpoint using the RESTful Services Utility Command

Starting with Oracle Key Vault release 21.10, you can list the endpoints that were active or inactive recently using the RESTful services utility.

To list the endpoint that were recently active, use the following command:

- `okv admin endpoint list --last-active-duration <last-active-duration>`

To list the endpoint with no recent activity, use the following command:

- `okv admin endpoint list --last-inactive-duration <last-inactive-duration>`

You can use the commands to check how often the endpoints connect to the Oracle Key Vault server. You can also use the commands to identify the defunct endpoints for house-keeping.

Changes for Oracle Key Vault Release 21.9

Oracle Key Vault release 21.9 introduces several new features.

- List the Wallet Membership of an Object Using RESTful Services Utility Command
  Starting with Oracle Key Vault release 21.9, you can now list all the wallet memberships of a given managed object.
- Allow List for Approved RESTful Connections
  Starting with Oracle Key Vault release 21.9, you can enable access to RESTful services utility from the allowed IP addresses only.

List the Wallet Membership of an Object Using RESTful Services Utility Command

Starting with Oracle Key Vault release 21.9, you can now list all the wallet memberships of a given managed object.

Following command is added to RESTful services utility to support this enhancement:

```
okv manage-access wallet list-object-wallets --uuid <uuid>
```

The RESTful services utility command is run by a user. Only those wallets that this user has access to, will be listed. The object may be a member of the wallets that the user running the command has no access to. These wallets are not listed.

Allow List for Approved RESTful Connections

Starting with Oracle Key Vault release 21.9, you can enable access to RESTful services utility from the allowed IP addresses only.

With this new feature, only the configured list of IP addresses can use the RESTful services utility. In earlier Oracle Key Vault releases, you could either enable or disable the RESTful services utility only.

Changes for Oracle Key Vault Release 21.8

Oracle Key Vault release 21.8 introduces several new features.

- Server-side Filtering for RESTful Services Utility Commands
  Starting in Oracle Key Vault release 21.8, you can now specify options to do server-side filtering for the RESTful services utility commands that list endpoints or wallets, list objects that endpoints have access to, list objects in wallet and for those that list completed backups.
- RESTful Services Utility Commands Support for Custom Attributes
  Starting with Oracle Key Vault Release 21.8, you can specify custom-attributes and KMIP-attributes of security objects as command line options when using RESTful services utility commands, such as `add`, `modify`, `delete`, and `get`. The `fetch` and `locate` commands also support additional attributes on the command line.

Server-side Filtering for RESTful Services Utility Commands

Starting in Oracle Key Vault release 21.8, you can now specify options to do server-side filtering for the RESTful services utility commands that list endpoints or wallets, list objects that endpoints have access to, list objects in wallet and for those that list completed backups.

You can filter the list of endpoints by platform, type, or registration status. You can filter the list of wallets by their type, either general or SSH server wallets. You can filter the list of objects that endpoints have access to or list objects in the wallet by type such as, secret or certificate, or state like active or compromised. You can filter the list of completed backups for a specific backup destination or filter them by type, that is, one-time or periodic, or simply filter by the backup name. You can specify more than one option for filtering and can also specify more than one value for the filtering option. For example, you can list all endpoints on Linux and Microsoft Windows platforms by using the following command with the filter options:

```
--platform "LINUX64, WINDOWS"
```

The following commands support this enhancement:

- `okv admin endpoint list`
- `okv admin endpoint list-objects`
- `okv manage-access wallet list`
- `okv manage-access wallet list-objects`
- `okv backup history list`

RESTful Services Utility Commands Support for Custom Attributes

Starting with Oracle Key Vault Release 21.8, you can specify custom-attributes and KMIP-attributes of security objects as command line options when using RESTful services utility commands, such as `add`, `modify`, `delete`, and `get`. The `fetch` and `locate` commands also support additional attributes on the command line.

KMIP attributes like `activation date` and `deactivation date` are now available as command line options `--activation-date` and `--deactivation-date` respectively. You can pass the custom-attributes using the new command line option `--custom-attribute`.

The following commands support this enhancement:

- `okv managed-object attribute add`
- `okv managed-object attribute modify`
- `okv managed-object attribute delete`
- `okv managed-object attribute get`
- `okv managed-object custom-attribute add`
- `okv managed-object custom-attribute modify`
- `okv managed-object custom-attribute delete`
- `okv managed-object custom-attribute get`
- `okv managed-object object locate`
- `okv managed-object object fetch`

Changes for Oracle Key Vault Release 21.7

Oracle Key Vault release 21.7 introduces several new features.

- RESTful Services Utility Changes to Support SSH Keys Management
  Starting with release 21.7, you can use the Oracle Key Vault RESTful services utility to create and register SSH keys and manage SSH Server wallets and SSH Server endpoints.

RESTful Services Utility Changes to Support SSH Keys Management

Starting with release 21.7, you can use the Oracle Key Vault RESTful services utility to create and register SSH keys and manage SSH Server wallets and SSH Server endpoints.

The following Oracle Key Vault RESTful services utility commands have been updated to support the SSH key pair creation and registration of SSH private and public keys:

- `okv managed-object key-pair create`
- `okv managed-object private-key register`
- `okv managed-object public-key register`

A new option **--ssh-user** is added to these commands. Use of this option makes the underlying public and private key objects identified as the SSH keys.

To support the creation of SSH Server endpoint and SSH Server wallet, following commands have been updated:

- `okv admin endpoint create`
- `okv manage-access wallet create`

Changes for Oracle Key Vault Release 21.6

Oracle Key Vault release 21.6 introduces several new features.

- Endpoint IP Address Attribute Added to endpoint get RESTful Command
  Oracle Key Vault supports endpoint IP address in the `endpoint get` RESTful command.
- Sign and Verify Operations in Oracle Key Vault
  Starting with Oracle Key Vault release 21.6, sign and verify operations can be performed using Oracle Key Vault's RESTful services, or the Oracle Key Vault client tool `okvutil`:

Endpoint IP Address Attribute Added to endpoint get RESTful Command

Oracle Key Vault supports endpoint IP address in the `endpoint get` RESTful command.

The endpoint IP address that was used at enrollment time is now recorded, and displayed with the `okv admin endpoint get --endpoint` *`endpoint_name`* command.

Sign and Verify Operations in Oracle Key Vault

Starting with Oracle Key Vault release 21.6, sign and verify operations can be performed using Oracle Key Vault's RESTful services, or the Oracle Key Vault client tool `okvutil`:

Both of the Oracle Key Vault RESTful API and Oracle Key Vault client utility `okvutil` provide sign and verify functionality.

The new or updated commands are as follows:

- `okv crypto data sign`
- `okv crypto data sign-verify`
- `okv crypto data sign`
- `okv crypto data sign-verify`
- `okvutil sign`
- `okvutil sign-verify`

Changes for Oracle Key Vault Release 21.5

Oracle Key Vault release 21.5 introduces several new features that affect this guide.

- Support for Cluster Management and Monitoring using RESTful Services Utility
  Starting in Oracle Key Vault release 21.5, you can deploy, manage, and monitor the multi-master cluster using RESTful services utility.
- Support for System Resources Monitoring using RESTful Services Utility
  Starting in Oracle Key Vault release 21.5, you can obtain the current and historical utilization metrics of the system resources such as CPU and memory using RESTful services utility. These system metrics would help you appropriately configure system resources for the Oracle Key Vault servers to meet the performance and scalability requirements of your deployment.
- RESTful Services Utility Commands Reduce Need for Intermediate JSON Files
  Starting in Oracle Key Vault release 21.5, you can specify custom-attributes and certain KMIP attributes as the command line options when using RESTful services utility to create, register, fetch and locate security objects.
- Support for Text Output Format in RESTful Services Utility
  Starting in Oracle Key Vault release 21.5, several RESTful services utility commands are enhanced to support the output in the **text** format.

Support for Cluster Management and Monitoring using RESTful Services Utility

Starting in Oracle Key Vault release 21.5, you can deploy, manage, and monitor the multi-master cluster using RESTful services utility.

Using the RESTful Services Utility, you can now perform several cluster management operations including creating a cluster, adding or deleting a node, enabling or disabling a node. You can also monitor and manage the cluster services and replication links between nodes using RESTful services utility.

The new commands are as follows:

- `okv cluster node create`
- `okv cluster node status`
- `okv cluster node add`
- `okv cluster node abort-pairing`
- `okv cluster node enable`

- `okv cluster node disable`
- `okv cluster node cancel-disable`
- `okv cluster node update`
- `okv cluster service start`
- `okv cluster service stop`
- `okv cluster service monitor`
- `okv cluster link enable`
- `okv cluster link disable`
- `okv cluster link monitor`

Related Topics

- Cluster Management Commands

Support for System Resources Monitoring using RESTful Services Utility

Starting in Oracle Key Vault release 21.5, you can obtain the current and historical utilization metrics of the system resources such as CPU and memory using RESTful services utility. These system metrics would help you appropriately configure system resources for the Oracle Key Vault servers to meet the performance and scalability requirements of your deployment.

Using the RESTful services utility, you can obtain the information about the:

- Configured system resources (CPU and memory)
- CPU and memory utilization metrics over a specified period, including load averages

The new or updated commands are as follows:

- `okv metrics server get`
- `okv server status get`
- `okv server info get`

Related Topics

- Monitoring Commands

RESTful Services Utility Commands Reduce Need for Intermediate JSON Files

Starting in Oracle Key Vault release 21.5, you can specify custom-attributes and certain KMIP attributes as the command line options when using RESTful services utility to create, register, fetch and locate security objects.

In earlier releases, commands that use the attributes or custom-attributes could only be executed using the JSON input method only. The RESTful services utility is enhanced to support the passing of attributes and custom-attributes as the command line options for the commands to create or register security objects. These commands also support simplified variants of the complex input.

The KMIP attributes "activation date" and "deactivation date" are exposed as the command line options `--activation-date` and `--deactivation-date` respectively. You can pass the custom-

attributes using the new command line option --custom-attribute. Several RESTful services utility commands also support simplified and complex format on name and custom attribute.

The following commands have been updated to accommodate this enhancement:

- `okv managed-object key create`
- `okv managed-object key register`
- `okv managed-object secret register`
- `okv managed-object certificate register`
- `okv managed-object certificate-request register`
- `okv managed-object opaque register`
- `okv managed-object public-key register`
- `okv managed-object private-key register`
- `okv managed-object object fetch`
- `okv managed-object object locate`

Related Topics

- Security Object Commands

Support for Text Output Format in RESTful Services Utility

Starting in Oracle Key Vault release 21.5, several RESTful services utility commands are enhanced to support the output in the **text** format.

In previous releases, the RESTful services utility commands always produced output in the JSON format. Now, you can use the new command line option **–output_format** to generate the command output in the text format. The **text** output format helps simplify the creation of automation scripts such as when the output of a command serves as input for another command.

Supported values for the --output_format option are:

- json (default value)
- text

The following commands have been updated to accommodate this enhancement:

- `okv managed-object certificate get`
- `okv managed-object certificate register`
- `okv managed-object certificate-request get`
- `okv managed-object certificate-request register`
- `okv managed-object key create`
- `okv managed-object key get`
- `okv managed-object key register`
- `okv managed-object object activate`
- `okv managed-object object destroy`
- `okv managed-object object locate`

- `okv managed-object object revoke`
- `okv managed-object opaque get`
- `okv managed-object private-key register`
- `okv managed-object public-key get`
- `okv managed-object public-key register`
- `okv managed-object secret get`
- `okv managed-object secret register`
- `okv managed-object wallet add-member`
- `okv managed-object wallet delete-member`
- `okv managed-object wallet list`

Related Topics

- Security Object Commands

Changes for Oracle Key Vault Release 21.4

Oracle Key Vault release 21.4 introduces several new features that affect this guide.

- RESTful Services Utility Commands to Support the Extractable Attribute for Symmetric Encryption Keys
  Starting in Oracle Key Vault release 21.4, to strengthen the protection of symmetric keys, you now can restrict these keys from leaving Oracle Key Vault by setting the extractable attribute.
- Support for Cryptographic Operations in RESTful Services Utility
  Oracle Key Vault release 21.4 adds the support for performing cryptographic operations within Oracle Key Vault.
- Support for Policy Based Automatic Purging of Old Oracle Key Vault Backups in RESTful Services Utility
  Starting in Oracle Key Vault release 21.4, you can create a policy to schedule the removal of one or more remote backups.
- Enhancements to Endpoint, Endpoint Group, and Wallet-Related RESTful Services Utility Commands
  Starting in Oracle Key Vault release 21.4, additional commands are available to enable you to perform more operations with endpoints, endpoint groups, and wallets.
- Support Endpoint Configuration Using the RESTful Services Utility
  Starting in Oracle Key Vault release 21.4, you can update the endpoint configuration parameters and endpoint settings for keys and secrets of an endpoint using the RESTful service utility command `okv admin endpoint update`.
- RESTful Commands to Set Date and Time Accommodate ISO 8601 Standard
  Starting in Oracle Key Vault release 21.4, the *`duration`* time interval settings will follow a subset of the ISO 8601 standard, and the fixed format for date and time settings are compatible with ISO 8601 when using RESTful commands.
- Support for Command Line Help for the RESTful Services Utility
  Starting in Oracle Key Vault release 21.4, you can find the command line help information about the RESTful services utility commands.

RESTful Services Utility Commands to Support the Extractable Attribute for Symmetric Encryption Keys

Starting in Oracle Key Vault release 21.4, to strengthen the protection of symmetric keys, you now can restrict these keys from leaving Oracle Key Vault by setting the extractable attribute.

The following commands have been updated to accommodate this enhancement:

- `okv managed-object attribute get`
- `okv managed-object attribute get-all`
- `okv managed-object attribute list`
- `okv managed-object attribute modify`
- `okv managed-object key create`
- `okv managed-object key register`
- `okv managed-object object locate`

Related Topics

- Access Management Commands
  You can use the access management commands to manage wallets and endpoint groups.

Support for Cryptographic Operations in RESTful Services Utility

Oracle Key Vault release 21.4 adds the support for performing cryptographic operations within Oracle Key Vault.

You can use either RESTful services utility commands or C and Java SDK to perform encryption and decryption operations.

This enhancement accommodates the use of symmetric keys that have been configured to not be extracted from Oracle Key Vault.

The new commands are as follows:

- `okv crypto data decrypt`
- `okv crypto data encrypt`

Related Topics

- Support for Cryptographic Operations in RESTful Services Utility
  Oracle Key Vault release 21.4 adds the support for performing cryptographic operations within Oracle Key Vault.

Support for Policy Based Automatic Purging of Old Oracle Key Vault Backups in RESTful Services Utility

Starting in Oracle Key Vault release 21.4, you can create a policy to schedule the removal of one or more remote backups.

The following commands have been updated:

- `okv backup destination create`

- `okv backup destination update`

The following commands are new:

- `okv backup destination delete-backup`
- `okv backup destination-policy create`
- `okv backup destination-policy delete`
- `okv backup destination-policy get`
- `okv backup destination-policy list`
- `okv backup destination-policy list-purged-backups`
- `okv backup destination-policy update`
- `okv backup destination resume-policy`
- `okv backup destination suspend-policy`

Related Topics

- Backup, Schedule, and Restore Commands
  You can use the backup, schedule, and restore commands to automate Oracle Key Vault appliance backups.

Enhancements to Endpoint, Endpoint Group, and Wallet-Related RESTful Services Utility Commands

Starting in Oracle Key Vault release 21.4, additional commands are available to enable you to perform more operations with endpoints, endpoint groups, and wallets.

The new commands are as follows:

- `okv admin endpoint get`
- `okv admin endpoint list`
- `okv admin endpoint list-objects`
- `okv admin endpoint resume`
- `okv admin endpoint suspend`
- `okv manage-access endpoint-group get`
- `okv manage-access endpoint-group list`
- `okv manage-access wallet add-object`
- `okv manage-access wallet get`
- `okv manage-access wallet list`
- `okv manage-access wallet list-objects`
- `okv manage-access wallet remove-object`

The commands to list objects for an endpoint (`okv admin endpoint list-objects`) and a wallet (`okv admin wallet list-objects`) provide an option to show or hide the wallet membership of the objects. Omitting wallet membership information of objects can improve command's performance.

Related Topics

- Administration Commands
  You can use the administration commands to manage client wallets and endpoints.
- Access Management Commands
  You can use the access management commands to manage wallets and endpoint groups.

Support Endpoint Configuration Using the RESTful Services Utility

Starting in Oracle Key Vault release 21.4, you can update the endpoint configuration parameters and endpoint settings for keys and secrets of an endpoint using the RESTful service utility command `okv admin endpoint update`.

The endpoint configuration parameters includes various PKCS#11 settings and endpoint settings for keys and secrets includes the `extractable` attribute setting for the new symmetric keys.

Related Topics

- okv admin endpoint update
  The `okv admin endpoint update` command updates the settings of an endpoint.
- RESTful Services Utility Commands to Support the Extractable Attribute for Symmetric Encryption Keys
  Starting in Oracle Key Vault release 21.4, to strengthen the protection of symmetric keys, you now can restrict these keys from leaving Oracle Key Vault by setting the extractable attribute.

RESTful Commands to Set Date and Time Accommodate ISO 8601 Standard

Starting in Oracle Key Vault release 21.4, the *`duration`* time interval settings will follow a subset of the ISO 8601 standard, and the fixed format for date and time settings are compatible with ISO 8601 when using RESTful commands.

You can specify the following formats:

- *`duration`* (follows a subset of the ISO 8601 standard)
- *`timestamp`* (is in a format that is compatible with the ISO 8601 standard)
- `now` (represents the current time when a command is run)

You can use these formats in the following combinations:

- *`timestamp`*
- `now`
- *`timestamp`* + *`duration`*
- `now` + *`duration`*

The *`timestamp`* format that has been used in previous releases is still supported.

The following commands have been updated for this enhancement:

- `okv backup schedule create`
- `okv backup schedule update`
- `okv managed-object attribute add`

- `okv managed-object attribute delete`
- `okv managed-object attribute modify`
- `okv managed-object certificate-request register`
- `okv managed-object key register`
- `okv managed-object object locate`
- `okv managed-object opaque register`
- `okv managed-object private_key register`
- `okv managed-object public-key register`
- `okv managed-object secret register`

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.
- Access Management Commands
  You can use the access management commands to manage wallets and endpoint groups.
- Backup, Schedule, and Restore Commands
  You can use the backup, schedule, and restore commands to automate Oracle Key Vault appliance backups.

Support for Command Line Help for the RESTful Services Utility

Starting in Oracle Key Vault release 21.4, you can find the command line help information about the RESTful services utility commands.

This enhancement enables you to find the detailed help information about the various categories, resources, and actions that are supported for all Oracle Key Vault RESTful services utility commands. The help information shows the command's syntax, and definitions for the available categories, resources, and actions as well as the configuration parameters that are applicable to all the commands.

Related Topics

- Getting Help Information for RESTful Services Utility Commands
  At the command line (but not in JSON), you can find detailed help information for each component of an `okv` RESTful services utility command.

Changes for Oracle Key Vault Release 21.3

Oracle Key Vault release 21.3 introduces one new feature that affects this guide.

- Enhancements for RESTful Services Utility Commands Used for Registration
  In Oracle Key Vault release 21.3, RESTful services utility commands that are used for the registration of managed objects will have additional attributes.

Enhancements for RESTful Services Utility Commands Used for Registration

In Oracle Key Vault release 21.3, RESTful services utility commands that are used for the registration of managed objects will have additional attributes.

The affected commands are as follows:

- `okv managed-object certificate register`
- `okv managed-object certificate-request register`
- `okv managed-object key register`
- `okv managed-object opaque register`
- `okv managed-object private-key register`
- `okv managed-object public-key register`
- `okv managed-object secret register`

In previous releases, these commands provided two attributes, `name` and `contactInfo`. In this release, in addition to these two attributes, the following new attributes are included:

- `activationDate`
- `deactivationDate`
- `processStartDate`
- `protectStopDate`

Related Topics

- Security Object Commands
  Endpoints can make use of the security object commands to operate on the managed objects.

1

Introduction to Oracle Key Vault RESTful Services

The Oracle Key Vault RESTful services utility commands enable you to perform many Oracle Key Vault tasks, such as managing endpoints or performing backups, at the command line.

About Oracle Key Vault RESTful Services

The Oracle Key Vault tasks that you can automate using RESTful services include the management of endpoints, wallets, security objects, deployment operations, and backup operations.

Though the Oracle Key Vault management console user interface is sufficient for managing these features, the process of completing these tasks is a manual one, with Oracle Key Vault administrators having to click through the user interface. A large distributed enterprise deployment often requires automation through scripting to enable mass deployment. The Oracle Key Vault RESTful services utility commands enable you perform all of these tasks in a way that facilitates faster deployment with less human intervention.

With Oracle Key Vault RESTful services, you can run a single service command from the command line. For most of the Oracle Key Vault RESTful services utility commands, you can specify command line options as a JavaScript Object Notation (JSON) input file. The reference sections in this guide provide examples of generating and modifying JSON input template for each command. The output of the RESTful services utility commands is in JSON format. To run the service commands from the command line, you will need to set certain configuration parameters. You can simplify the execution of RESTful services utility commands by having these commonly used parameters in the RESTful services configuration file. These parameters cover areas that are universal, such as the name of the RESTful administrator who needs to run the command. Oracle Key Vault also provides a logging properties file to customize how logging is handled. In order to run the RESTful service utility, the endpoint must have at minimum Java Runtime Environment version 1.7.0.21 installed.

After you use RESTful services to perform Oracle Key Vault tasks, you should disable the RESTful services to minimize the number of entry points to Oracle Key Vault.

General Process for Using Oracle Key Vault RESTful Services

After you enable the RESTful services, in some cases, you will use JSON to perform the Oracle Key Vault RESTful services tasks.

To configure the Oracle Key Vault RESTful services, you will follow these general steps:

1. Enable RESTful services from the Oracle Key Vault management console.

   This step entails ensuring that the endpoint meets the system requirements, and then using the Oracle Key Vault management console to enable the network services and the RESTful services functionality.

2. Download the RESTful service utility `okvrestclipackage.zip`.

   This file contains an `okvrestcli.jar` file, the RESTful services command line utility script, a configuration file, and the default logging file.

3. Customize the following configuration and logging files to work with your environment:

   - `okvrestcli.ini` contains properties that are specific to your environment, such as the name of the user who will run the RESTful services utility commands.
   - `okvrestcli_logging.properties` determines how logging is handled.

After the Oracle Key Vault RESTful services have been configured, you can begin to use the RESTful services utility commands right away. You can run the commands individually, using different methods. In most cases, the RESTful services utility commands support JSON formatting.

Related Topics

- Running RESTful Services Utility Commands Using the JSON Syntax
  The RESTful services utility commands support JSON syntax, and after you have generated the JSON output, you can use it in combination with a command line execution of the command.

Required Privileges for Using RESTful Services

The required RESTful services privileges are consistent with the privileges required to perform the same task in the Oracle Key Vault management console.

Based on the activity that you want to perform, the required privileges are as follows:

- **Creating endpoints:** System Administrator role or the Create Endpoint system privilege
- **Managing endpoints:** System Administrator role or the Manage Endpoint object privilege for the endpoint
- **Creating endpoint groups:** Key Administrator role or the Create Endpoint Group system privilege
- **Managing endpoint groups:** Key Administrator role or the Manage Endpoint Group object privilege for the endpoint group
- **Managing wallets and keys:** Key Administrator role or wallet privileges
  There are three modes for wallet privileges:
  - Read-only access (`RO`)
  - Read-and-modify access (`RM`)
  - Manage-wallet access (`MW`)

You can grant wallet privileges in any of the following combinations:

  - `RO`
  - `RM`
  - `RO_MW`
  - `RM_MW`

  For example, if an endpoint is assigned only read-only (`RO`) and read-and-modify (`RM`) wallet access, then you cannot use the `okv managed-object wallet add-member` on the endpoint because this command requires manage-wallet access (`RM_MW`).

- **Managing security objects:** Key Administrator role
- **Executing commands to check the status of and information about clusters or primary-standby deployments:** System Administrator role
- **Managing Backup and Restore:** System Administrator Role

To simplify administration tasks, you can create a user who has one or more of these roles. Typically, this user is an administrator who must self-register their databases with Oracle Key Vault by using scripts that will need to perform the actions that need these privileges.

You do not need to have endpoint administrator privileges to use the Oracle Key Vault RESTful services.

2

Getting Started with Oracle Key Vault RESTful Services

After you download the RESTful services utility and customize its configuration files, you can begin to use the Oracle Key Vault RESTful services utility commands.

- Enabling or Disabling Oracle Key Vault RESTful Services
  You enable and disable RESTful services from the Oracle Key Vault management console.
- Oracle Key Vault RESTful Services Configuration and Logging Files
  Oracle Key Vault provides two files, `okvrestcli.ini` and `okvrestcli_logging.properties`, that you can use to specify required or optional settings for when you run RESTful services utility commands.
- Getting Help Information for RESTful Services Utility Commands
  At the command line (but not in JSON), you can find detailed help information for each component of an `okv` RESTful services utility command.
- Running Oracle Key Vault RESTful Services Utility Commands
  Oracle Key Vault provides a variety of ways to run RESTful services utility commands.
- Naming Guidelines for Objects
  The naming guidelines affect the following Oracle Key Vault objects: users, user groups, endpoints, endpoint groups, and virtual wallets.
- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.
- Using RESTful Services with LDAP Users
  Both regular Oracle Key Vault administrators and properly authorized LDAP users can log in to a server to run Oracle Key Vault RESTful services utility commands.

Enabling or Disabling Oracle Key Vault RESTful Services

You enable and disable RESTful services from the Oracle Key Vault management console.

- Enabling RESTful Services
  After checking the endpoint requirements, and enabling network services, you can enable RESTful services, download the RESTful software utility, and then customize its configuration files.
- Disabling RESTful Services
  You should enable RESTful services for short periods during when administrative tasks are performed.

Enabling RESTful Services

After checking the endpoint requirements, and enabling network services, you can enable RESTful services, download the RESTful software utility, and then customize its configuration files.

- Step 1: Check the Endpoint System Requirements
  Before you can provision endpoints with the RESTful command-line interface, you must have the tools to transfer data securely across the network.
- Step 2: Enable Network Services
  You must configure web access for RESTful clients by their IP addresses to access the Oracle Key Vault server.
- Step 3: Enable RESTful Services
  After you enable the network services, you can enable the RESTful services.
- Step 4: Download the RESTful Services Utility
  The RESTful services utility is in the `okvrestclipackage.zip` file.
- Step 5: Configure the RESTful Services Utility
  After you have downloaded the RESTful services utility, you must modify a couple of files that are included in its zip file.

Step 1: Check the Endpoint System Requirements

Before you can provision endpoints with the RESTful command-line interface, you must have the tools to transfer data securely across the network.

1. Log in to the endpoint host as an endpoint administrator.
2. Ensure that you have the following tools:
   - OpenSSL 1.0.1p or later
   - JAVA 8 or later. If you plan to deploy RESTful services on a database server with Oracle Database release 12.2.0.1 or later, then you can use the embedded Java Runtime Environment (JRE) in `$ORACLE_HOME/jdk/jre`.
     For database installations from Oracle release 12.2.0.1 and later, set `JAVA_HOME` to `$ORACLE_HOME/jdk/jre`, and add `JAVA_HOME/bin` to the `PATH`. For earlier database releases, download and install JAVA 8 or later, and then set `JAVA_HOME` and `PATH` appropriately. OpenJDK is not supported.

Step 2: Enable Network Services

You must configure web access for RESTful clients by their IP addresses to access the Oracle Key Vault server.

You can allow all IP addresses or restrict access to a subset of IP addresses that you designate in this step. Note, that this option will also restrict access to the Oracle Key Vault management console.

1. Log in to the Oracle Key Vault management console as a user with the System Administrator role.
2. Select **System**, then **Settings** from the left sidebar.

   The Settings page appears. Go to the Network Details section.
3. For **Web Access** select *one* of the IP address options for the RESTful client:
   - **All** to allow all IP addresses.
   - **IP address(es)** to designate a set of IP addresses. After you select this option, enter the IP addresses in the next field, separating each IP address by a space.
4. Click **Save**.

Step 3: Enable RESTful Services

After you enable the network services, you can enable the RESTful services.

In a multi-master cluster environment, enabling RESTful services on one node will enable it for the entire cluster.

1. Log in to the Oracle Key Vault management console as a user with the System Administrator role.
2. Select **System**, then **Settings** from the left sidebar.

   The Settings page appears. Go to the **System Configuration** section, then to the **RESTful Services** section within it.
3. From the options, select **IP address(es)** or **All**.

   > **Note**
   >
   > If you select the **IP address(es)** option, then provide the allowed list of IP address(es) in the text box.

4. Click **Save**.

Step 4: Download the RESTful Services Utility

The RESTful services utility is in the `okvrestclipackage.zip` file.

In addition to the new RESTful service utility introduced in the Oracle Key Vault 21.1 release, Oracle Key Vault continues to support earlier implementation of the RESTful service utility that you can download as Classic RESTful Service Utility.

- Use one of the following methods to download the Oracle Key Vault RESTful services utility `okvrestclipackage.zip` file:
  - From the Home page of the Oracle Key Vault management console:
    1. Log in as a user with the System Administrator role.
    2. Select the **System** tab.
    3. In the left sidebar, select **Settings**.
    4. Under System Configuration, select **RESTful Services**.
    5. In the RESTful Services dialog box, select **Download**.
       To download the Oracle Key Vault Classic RESTful Service Utility, click **Download Classic Utility**. For information about using this version, see the release 18.6 version of *Oracle Key Vault Administrator's Guide*
    6. In the Opening okvrestclipackage.zip dialog box, select **Save** to save the `okvrestclipackage.zip` file locally.
  - From the Endpoint Enrollment and Software Download of the Oracle Key Vault management console:
    1. Connect to the Oracle Key Vault management console.
       The login page to the Oracle Key Vault management console appears. ***Do not log in.***
    2. In the lower-right corner of the login page under **Login**, click **Endpoint Enrollment and Software Download**.

3. Click the **Download RESTful Service Utility** tab.
4. Click the **Download** button.
   To download the Classic RESTful Service Utility, click **Download Classic Utility**.
5. Download the `okvrestclipackage.zip` to a secure location.

– Using a command-line HTTP client such as `wget` or `curl`. In a primary-standby configuration, enter the IP address of the primary database. For example:

```
wget --no-check-certificate https://ip_address:5695/okvrestclipackage.zip
curl -k https://ip_address:5695/okvrestclipackage.zip -o okvrestclipackage.zip
curl -O -k https://ip_address:5695/okvrestclipackage.zip
```

Step 5: Configure the RESTful Services Utility

After you have downloaded the RESTful services utility, you must modify a couple of files that are included in its zip file.

1. Move the `okvrestclipackage.zip` file to the Oracle Key Vault home directory (`OKV_HOME`) in the endpoint.

   You can move the zip file to any secure location, but having it in the Oracle Key Vault home directory in the endpoint is convenient for managing the Oracle Key Vault RESTful files in a central location. This guide assumes that you downloaded the zip file onto the endpoint.
2. Unzip the `okvrestclipackage.zip` file.

   For example:

   ```
   unzip okvrestclipackage.zip
   ```

   The following directory structure is created:

   - Directory where you placed the `okvrestclipackage.zip` file, such as the `OKV_HOME` directory
     – `bin`
       * `okv`
       * `okv.bat`
     – `lib`
       * `okvrestcli.jar`
     – `conf`
       * `okvrestcli.ini`
       * `okvrestcli_logging.properties`
3. In the RESTful services command line utility script, set the `OKV_RESTCLI_CONFIG` variable.

   `OKV_RESTCLI_CONFIG` sets the location of the `okvrestcli.ini` configuration file. The RESTful services command line utility script for Linux platforms is `okv` and the utility script for Microsoft Windows is `okv.bat`.
4. Set the `JAVA_HOME` environment variable to a Java Runtime Environment (JRE) installation with the minimum version 1.7.0.21.

Next, you are ready to modify the `okvrestcli.ini` and `okvrestcli_logging.properties` configuration files for your environment.

Related Topics

- Oracle Key Vault RESTful Services Configuration and Logging Files
  Oracle Key Vault provides two files, `okvrestcli.ini` and `okvrestcli_logging.properties`, that you can use to specify required or optional settings for when you run RESTful services utility commands.

Disabling RESTful Services

You should enable RESTful services for short periods during when administrative tasks are performed.

RESTful Services are disabled by default. After you have performed administrative tasks using the RESTful services, you should disable RESTful services.

1. Log in to the Oracle Key Vault management console as a user with the System Administrator role.
2. Select **System**, then **Settings** from the left sidebar.

   The Settings page appears. Go to the System Configuration section, then to the RESTful Services section within it.
3. Un-check the box to the right of **Enable** in the **RESTful Services** section.
4. In the System Settings page, click the **Save**.

Oracle Key Vault RESTful Services Configuration and Logging Files

Oracle Key Vault provides two files, `okvrestcli.ini` and `okvrestcli_logging.properties`, that you can use to specify required or optional settings for when you run RESTful services utility commands.

- okvrestcli.ini Configuration File
  The `okvrestcli.ini` file enables you to control global settings that are used in the Oracle Key Vault RESTful services utility commands.
- okvrestcli_logging.properties Log File Parameter Settings
  The `okvrestcli_logging.properties` log file determines how logging is handled for Oracle Key Vault RESTful services activities.

okvrestcli.ini Configuration File

The `okvrestcli.ini` file enables you to control global settings that are used in the Oracle Key Vault RESTful services utility commands.

- About the okvrestcli.ini Configuration File
  The `okvrestcli.ini` file enables you to configure commonly used settings when you run Oracle Key Vault RESTful services utility commands.
- okvrestcli.ini Configuration Parameters
  The `okvrestcli.ini` parameters cover settings such as the name of a user, the location of the `okvclient.ora` file, and so on.

- [DEFAULT] and Named Profiles in the okvrestcli.ini File
  The `[DEFAULT]` and named profile sections of the `okvrestcli.ini` file enable you to maintain different sets of configuration parameter settings that can be applied when executing commands in different contexts.
- Precedence Order of okvrestcli.ini Parameters
  When you run an Oracle Key Vault RESTful service command, the configuration parameter values are determined on the basis of an order of precedence.
- Using an Alternative Configuration File
  You can use an alternative parameter configuration file from the `okvrestcli.ini` configuration file.

About the okvrestcli.ini Configuration File

The `okvrestcli.ini` file enables you to configure commonly used settings when you run Oracle Key Vault RESTful services utility commands.

These are settings such as the user's name or the IP address of an Oracle Key Vault server. The RESTful service utility requires these kinds of configuration parameters in the `okvrestcli.ini` file to be set for each command execution. The settings that you set in this file are automatically applied to all Oracle Key Vault RESTful services utility commands without the need for you to manually enter them at the command line each time that you want to run the command.

The configuration parameters in the `okvrestcli.ini` are grouped together in different sections called named profiles. Each section includes the profile name and list of the parameters that are associated with the profile. When you run the command with a named profile (using the `--profile` *`profile_name`* parameter), the configuration parameters listed under the named profile apply for the execution of the command. The configuration parameters listed under the `[DEFAULT]` profile represent default parameter settings that apply when either no named profile is specified or the parameter is not listed under the named profile.

By default, the `okvrestcli.ini` is in the same location where you downloaded the Oracle Key Vault RESTful services utility, in the `OKV_HOME/conf` directory of the endpoint.

Related Topics

- Using an Alternative Configuration File
  You can use an alternative parameter configuration file from the `okvrestcli.ini` configuration file.

okvrestcli.ini Configuration Parameters

The `okvrestcli.ini` parameters cover settings such as the name of a user, the location of the `okvclient.ora` file, and so on.

The `okvrestcli.ini` parameters are as follows:

- `server`: Determines the target Oracle Key Vault server where command is sent for execution. Enter the IP address of this server. Server information can also be obtained from the `okvclient.ora` file when you set the `okv_client_config` parameter.

  In a multi-master deployment of an Oracle Key Vault cluster, Oracle Key Vault dynamically updates the server information in the endpoint's configuration file `okvclient.ora` based on the endpoint's cluster subgroup setting, as well as any changes to the cluster topology or the state of the Oracle Key Vault cluster nodes. Using the server information from endpoint's `okvclient.ora` enables Oracle Key Vault to automatically select the best Oracle

Key Vault node to run the REST commands without you having to constantly update the `SERVER` parameter in `okvrestcli.ini`.

- `okv_client_config`: Specifies the full path of the `okvclient.ora` file for an endpoint. By default, this file is located in the `$OKV_HOME/conf` directory. This parameter is mandatory if you want to run managed-object RESTful services utility commands, which must always be run with the identity of an endpoint. The Oracle Key Vault RESTful services utility uses only the following information from the `okvclient.ora` file:
  - server information: IP address or host name of Oracle Key Vault server(s).
  - `SSL_WALLET_LOC`: location of the wallet that the endpoint uses.

  This is a string value and is required.
- `user`: Specifies the Oracle Key Vault user who is executing the RESTful services command. The user must have appropriate privileges to run the command.

  Oracle Key Vault does not use the `user` parameter when the RESTful services utility commands for the managed-object category are run. These commands are always run with the identity of an endpoint that is set with the `okv_client_config` parameter.
- `client_wallet`: Specifies the absolute path to a wallet which contains user credentials. This wallet can be used to log into the Oracle Key Vault server without having to manually specify the user's password. The user information is obtained from the `user` parameter. The `client_wallet` parameter enables implementation and use of automation scripts that need to run in an unattended mode.

  Oracle Key Vault does not use the `client_wallet` parameter when the RESTful services utility commands for the managed-object category are run. These commands are always executed with the identity of an endpoint that is set with the `okv_client_config` parameter.
- `log_property`: Specifies the full path of the Java logging property file. If this parameter is not set, then when you execute a RESTful command, Oracle Key Vault generates a log file with the default name in the current directory with the `INFO` level along with the message in a log file saying `log_property` is not configured. The default log property file is a part of the downloaded `okvrestclipackage.zip` file. This file enables you to customize the log file and its format. This is a string value and it is optional.

[DEFAULT] and Named Profiles in the okvrestcli.ini File

The `[DEFAULT]` and named profile sections of the `okvrestcli.ini` file enable you to maintain different sets of configuration parameter settings that can be applied when executing commands in different contexts.

The `okvrestcli.ini` file is organized as one or more named profile sections. A named profile section represents a collection of configuration parameter settings that are logically group together. A named profile section includes:

- Named profile section header denoted as [*`profile_name`*]
- Listing of configuration parameters under the name profile header

You apply the configuration parameter settings listed under a named profile by specifying the profile name in the command line with parameter `--profile` *`profile_name`*.

The `[DEFAULT]` profile lists the default values of the `okvrestcli.ini` parameters. The parameter settings under the `[DEFAULT]` profile apply when either no named profile is specified during command execution or the parameter is not listed under the named profile, and assuming you do not specify the parameter in the command line.

The following example shows the use of profiles that is suitable for connecting to Oracle Key Vault using identities of different endpoints. This is useful in an environment where you have isolated PDB endpoints configured on the same host. This `okvrestcli.ini` file has a named profile for each PDB endpoint that points to respective `okvclient.ora` file.

```
[DEFAULT]
log_property=/usr/local/okv/logging.property
server = 192.0.2.191

[HR_PDB]
okv_client_config=/usr/local/okv/hr_ep/okvclient.ora

[FIN_PDB]
okv_client_config=/usr/local/okv/finance_ep/okvclient.ora

[SALES_PDB]
okv_client_config=/usr/local/okv/sales_ep/okvclient.ora
```

To create a key using `HR_DB` endpoint, you use the `[HR_DB]` profile:

```
okv managed-object key create --profile HR_DB --algorithm AES --length 256 --
mask "ENCRYPT,DECRYPT" --wallet hr_wallet
```

This command uses the `okv_client_config` parameter from `[HR_DB]` profile. Other configuration parameters (for example, `log_property` and `server`) are applied from the `[DEFAULT]` profile.

This example shows the use case of using profiles in a multi-master cluster environment where you create a profile for each node in the cluster to use settings that are specific to that node. The following example contains profiles for three nodes in a cluster:

```
[DEFAULT]
log_property=/usr/local/okv/logging.property
user=okvadmin
server=192.0.2.191

[NODE1]
server=192.0.2.191

[NODE2]
server=192.0.2.192

[NODE3]
server=192.0.2.193
```

To run a command on `NODE2`, you use the `[NODE2]` profile:

```
okv server status get --profile NODE2
```

This command uses the server entry from `[NODE2]` profile. The other configuration parameter settings are used from the `[DEFAULT]` profile.

Before you work with `[DEFAULT]` settings and profiles, ensure that you understand the precedence order of the `okvrestcli.ini` parameters.

Related Topics

- Precedence Order of okvrestcli.ini Parameters
  When you run an Oracle Key Vault RESTful service command, the configuration parameter values are determined on the basis of an order of precedence.

Precedence Order of okvrestcli.ini Parameters

When you run an Oracle Key Vault RESTful service command, the configuration parameter values are determined on the basis of an order of precedence.

Parameter Precedence Order

The order of precedence for `okvrestcli.ini` configuration file parameters (except for the `server` entry) is as follows:

1. Parameter value is specified by the user in the command line.
2. Parameter value is specified in the profile section. User includes the `--profile` parameter in the command line.
3. Parameter value is specified in the `[DEFAULT]` profile. User makes no reference in the command line.

Examples of How Parameter Precedence Works

The following examples show how parameter precedence works for the following `okvrestcli.ini` file, which contains different settings for the `user` parameter under the `[DEFAULT]` and `[HR]` profiles.

```
[DEFAULT]
user= psmith

[HR]
user=jgreenberg
```

- **Example 1:** To specify the default user, `psmith`, simply omit any reference to this user from the command line.

  ```
  okv manage-access endpoint-group add-endpoint --endpoint-group epg_1 --
  endpoint ep_1
  ```

- **Example 2:** To override the default user and specify user `jgreenberg`, who is in the `HR` profile, specify the `HR` profile in the command line.

  ```
  okv manage-access endpoint-group add-endpoint --profile HR --endpoint-
  group epg_1 --endpoint ep_1
  ```

- **Example 3:** To override all the `user` settings in `okvrestcli.ini`, include the `--user` setting in the command line.

  ```
  okv manage-access endpoint-group add-endpoint --user kjones --endpoint-
  group epg_1 --endpoint ep_1
  ```

server Parameter Precedence Order

The `server` parameter has a slightly different precedence behavior from the other `okvrestcli.ini` parameter settings, because in addition to the `okvrestcli.ini` file, its setting can come from the `okvclient.ora` file. The location of the `okvclient.ora` file is specified with the `okv_client_config` parameter in `okvrestcli.ini`. The `server` entry that is specified directly takes precedence over the `server` entry from the `okv_client_config` parameter.

The order of precedence for the `server` entry is as follows:

1. `server` parameter value is specified by the user in the command line.
2. Server information is obtained from the `okvclient.ora` file. User specifies this file by including the `okv_client_config` parameter in the command line.
3. `server` parameter value is specified in the profile section. User includes the `--profile` parameter in the command line.
4. Server information is obtained from the `okvclient.ora` file, which is set by the `okv_client_config` parameter from a profile section. User specifies this profile by using the `--profile` parameter in the command line.
5. `server` parameter value is specified in the `[DEFAULT]` profile. User makes no reference in the command line.
6. Server information is obtained from the `okvclient.ora` file that is specified with `okv_client_config` parameter in the `[DEFAULT]` section. User makes no reference in the command line.

Examples of How the server Parameter Precedence Order Works

The following examples show how the `server` parameter precedence works based on various ways that this parameter can be set:

- **Example 1:** Assume that the `okvrestcli.ini` configuration file has the following setting:

  ```
  [DEFAULT]
  server=192.0.2.190
  ```

  To use this default setting (that is, to use IP address 192.0.2.190), omit any reference to it from the command line.

  ```
  okv manage-access endpoint-group add-endpoint --endpoint-group epg_1 --
  endpoint ep_1
  ```

- **Example 2:** Assume that the `okvrestcli.ini` configuration file has the following setting:

  ```
  [DEFAULT]
  okv_client_config=/usr/local/okv/okvclient/okvclient.ora
  ```

The `okv_client_config` parameter points to an `okvclient.ora` file that contains the `server` setting that you want to use. Because `okv_client_config` is in the `[DEFAULT]` section, to use this `okvclient.ora`, omit the reference to it from the command line.

```
okv manage-access endpoint-group add-endpoint --endpoint-group epg_1 --
endpoint ep_1
```

- **Example 3:** Assume that the `okvrestcli.ini` configuration file has the following settings for the default and for a profile called `[NODE_1]`:

  ```
  [DEFAULT]
  okv_client_config=/usr/local/okv/okvclient/okvclient.ora

  [NODE_1]
  server=192.0.2.191
  ```

  To override the default server setting from `okv_client_config` with the `[NODE_1]` profile setting of 192.0.2.191, include the `--profile` parameter in the command line.

  ```
  okv manage-access endpoint-group add-endpoint --profile node_1 --endpoint-
  group epg_1 --endpoint ep_1
  ```

- **Example 4:** Assume that the `okvrestcli.ini` configuration file has the following settings:

  ```
  [DEFAULT]
  server = 192.0.2.191

  [HR]
  okv_client_config=/usr/local/okv/hr_ep/okvclient.ora
  ```

  To override the default and use the server setting in the `okvclient.ora` file, as with Example 3, include the `--profile` parameter in the command.

  ```
  okv manage-access endpoint-group add-endpoint --profile hr --endpoint-
  group epg_1 --endpoint ep_1
  ```

- **Example: 5:** Assume that the `okvrestcli.ini` configuration file is as follows:

  ```
  [DEFAULT]
  server = 192.0.2.191

  [HR]
  okv_client_config=/usr/local/okv/hr_ep/okvclient.ora
  ```

  To override all of these settings, directly specify the appropriate server IP address setting in the command line.

  ```
  okv manage-access endpoint-group add-endpoint --server 192.0.2.192 --
  endpoint-group epg_1 --endpoint ep_1
  ```

This works with the `okv_client_config` parameter setting as well.

```
okv manage-access endpoint-group add-endpoint --okv_client_config /usr/
local/okv/okvclient/okvclient.ora --endpoint-group epg_1 --endpoint ep_1
```

The following example uses both a named profile (`HR`) and the `--server` parameter. The `--server` parameter overrides the `server` information from the `okvclient.ora` file specified in the `[HR]` profile.

```
okv managed-object key create --profile HR --server 192.0.2.192 --
algorithm AES --length 256 --mask "ENCRYPT,DECRYPT" --wallet hr_wallet
```

Using an Alternative Configuration File

You can use an alternative parameter configuration file from the `okvrestcli.ini` configuration file.

By default, Oracle Key Vault uses the `okvrestcli.ini` configuration file to control commonly used settings for the Oracle Key Vault RESTful services utility commands. You can create your own version of this configuration file and specify it in the command line execution.

- To use a different configuration file, include the `--config` parameter when you run the command. Add the `--config` parameter before the command-specific parameters, as follows:

  ```
  okv managed-object key create --config full_path_to_conf_file --algorithm
  AES --length 128 --mask "ENCRYPT,DECRYPT,EXPORT"
  ```

  Follow the same precedence rules that you would follow for the `okvrestcli.ini` file. For example, suppose the new configuration file has a profile that you want to use called `[HR]`. You would specify it as follows:

  ```
  okv managed-object key create --config full_path_to_conf_file --profile hr
  --algorithm AES --length 128 --mask "ENCRYPT,DECRYPT,EXPORT"
  ```

Related Topics

- okvrestcli.ini Configuration Parameters
  The `okvrestcli.ini` parameters cover settings such as the name of a user, the location of the `okvclient.ora` file, and so on.
- [DEFAULT] and Named Profiles in the okvrestcli.ini File
  The `[DEFAULT]` and named profile sections of the `okvrestcli.ini` file enable you to maintain different sets of configuration parameter settings that can be applied when executing commands in different contexts.
- Precedence Order of okvrestcli.ini Parameters
  When you run an Oracle Key Vault RESTful service command, the configuration parameter values are determined on the basis of an order of precedence.

okvrestcli_logging.properties Log File Parameter Settings

The `okvrestcli_logging.properties` log file determines how logging is handled for Oracle Key Vault RESTful services activities.

Modifying the `okvrestcli_logging.properties` is optional. If you do not configure it, then Oracle Key Vault creates and updates a default logging file when you run the RESTful services utility commands.

By default, the `okvrestcli_logging.properties` file is in the location you downloaded the Oracle Key Vault RESTful services utility, in the `OKV_HOME/conf` directory of the endpoint.

The parameter settings for `okvrestcli_logging.properties` are as follows:

- `java.util.logging.FileHandler.pattern` specifies one of the following patterns for generating the output file name. The default is `%h/java%u.log`.
  - `/` is the local path name separator.
  - `%h` is the value of the `user.home` system property.
  - `%g` is the generation number to distinguish rotated logs.
  - `%u` is a unique number to resolve conflicts.
  - `%%` translates to a single percent sign `%`.
- `java.util.logging.FileHandler.limit` specifies an approximate maximum amount to write (in bytes) to any one file. If this is zero, then there is no limit. The default is 200000.
- `java.util.logging.FileHandler.count` specifies how many output files to cycle through. The defaults is 5.
- `java.util.logging.FileHandler.formatter` specifies the name of a `Formatter` class to use. The default is `java.util.logging.XMLFormatter`.
- `java.util.logging.ConsoleHandler.level` specifies the default level for the handler. The default to `INFO`.
  The available logging levels are `ALL`, `TRACE`, `FINEST`, `FINER`, `FINE`, `CONFIG`, `INFO`, `WARNING`, `SEVERE`, and `OFF`.

  Any logging at `INFO` and above provides complete details. If you set the logging level to `SEVERE`, then you will only see messages with the `SEVERE` logging level, which generally correspond to serious problems. To diagnose the issue, you may need more details and that can be obtained with levels that produce more information, not just the occurrences of the serious issues.

An example of these settings is as follows:

```
handlers= java.util.logging.FileHandler

# default file output is in user's home directory.
java.util.logging.FileHandler.pattern = /usr/local/okv/okvrest.log
java.util.logging.FileHandler.limit = 200000
java.util.logging.FileHandler.count = 1
#java.util.logging.FileHandler.formatter = java.util.logging.XMLFormatter
java.util.logging.FileHandler.formatter = com.oracle.okv.rest.log.OkvFormatter

# Limit the message that are printed on the console to INFO and above.
java.util.logging.ConsoleHandler.level = FINER
#java.util.logging.ConsoleHandler.formatter = java.util.logging.XMLFormatter
java.util.logging.ConsoleHandler.formatter =
com.oracle.okv.rest.log.OkvFormatter
```

Getting Help Information for RESTful Services Utility Commands

At the command line (but not in JSON), you can find detailed help information for each component of an `okv` RESTful services utility command.

- okv category Component Help Information
  Entering `okv --help` returns help information for the `category` component of the `okv` command.
- okv resource Component Help Information
  Entering `okv *category* --help` returns detailed information about the `resource` components for the specified *`category`*.
- okv action Component Help Information
  Entering `okv *category resource* --help` returns detailed information about the `action` component of an `okv` command with the specified *`category`* and *`resource`*.
- okv option Component Help Information
  Entering `okv *category resource action* --help` returns detailed information about the `option` component of an `okv` command with the specified *`category`*, *`resource`*, and *`action`*.

okv category Component Help Information

Entering `okv --help` returns help information for the `category` component of the `okv` command.

Syntax: `okv --help`

Example Input: `okv --help`

Example Output:

```
Oracle Key Vault REST CLI Version 21.12.0.0.0 Built 10/13/2025 12:34
Usage: okv <category> <resource> <action> [<rest-cli-parameters>]
[<parameters>]

Command:
  <category> :
    managed-object                -     Commands that endpoints can execute
to manage security objects.
    backup                        -     Administration commands to manage
Oracle Key Vault appliance backup and restore.
    admin                         -     Administration commands to manage
client wallets and endpoints.
    manage-access                 -     Access management commands to manage
wallets and endpoint groups.
    server                        -     Monitoring commands to retrieve
static or dynamic information about an Oracle Key Vault server.
    cluster                       -     Monitoring commands to retrieve
static or dynamic information about a cluster or a cluster node.
    primary-standby               -     Monitoring commands to retrieve
static or dynamic information about the Oracle Key Vault primary-standby
configuration.
    crypto                        -     Commands to perform cryptographic
operations.
    metrics                       -     Commands to perform metrics
```

```
operations.
    util                          -     Command to perform json parser
operations.

<rest-cli-parameters>:
    --client_wallet <arg>       Client wallet.
    --config <arg>              OKV REST CLI configuration file
                                (okvrestcli.ini) location.
    --from-json <arg>           Input file in JSON.
    --generate-json-input       Generate JSON input template file.
    --help                      List available options.
    --okv_client_config <arg>   OKV Client configuration file
                                (okvclient.ora) location.
    --output_format <arg>       Command output format, 'text' or 'json
    --profile <arg>             Profile name in configuration file
                                (okvrestcli.ini).
    --server <arg>              OKV server IP address or hostname.
    --user <arg>                Username.
    --version <arg>             Version for backward compatibility.
```

`rest-cli-parameters` shows the list of parameters that apply to all commands.

okv resource Component Help Information

Entering `okv` *`category`* `--help` returns detailed information about the `resource` components for the specified *`category`*.

Syntax: `okv` *`category`* `--help`

Example Input: `okv admin --help`

Example Output:

```
Oracle Key Vault REST CLI Version 21.12.0.0.0 Built 10/13/2025 12:34
Usage: okv admin <resource> <action> [<rest-cli-parameters>] [<parameters>]

Command:
  <category> : admin
  <resource> :
    endpoint                    -     Commands to manage
endpoints.
    client-wallet               -     Commands to manage client wallets.
```

okv action Component Help Information

Entering `okv` *`category resource`* `--help` returns detailed information about the `action` component of an `okv` command with the specified *`category`* and *`resource`*.

Syntax: `okv` *`category resource`* `--help`

Example Input: `okv admin endpoint --help`

Example Output:

```
Oracle Key Vault REST CLI Version 21.12.0.0.0 Built 10/13/2025 12:34
Usage: okv admin endpoint <action> [<rest-cli-parameters>] [<parameters>]
```

```
Command:
  <category> : admin
  <resource> : endpoint
  <action> :
    create                          -     Add a new endpoint to the Oracle Key
Vault.
    delete                          -     Remove an endpoint from the Oracle
Key Vault.
    update                          -     Update the settings of an
endpoint.
    check-status                    -     Display the current state of an
endpoint. The state will be either ACTIVE or PENDING.
    get                             -     Retrieve information about an
endpoint.
    list                            -     Display a list of
endpoints.
    download                        -     Download the endpoint software
(okvclient.jar) to the specified directory.
    provision                       -     Download and install the endpoint
software in the specified directory.
    re-enroll                       -     Re-enroll a previously enrolled
endpoint.
    suspend                         -     Suspend an
endpoint.
    resume                          -     Resume an
endpoint.
    get-enrollment-token            -     Retrieve an enrollment token for a
registered endpoint.
    re-enroll-all                   -     Re-enroll all previously enrolled
endpoints.
```

okv option Component Help Information

Entering `okv` *`category resource action`* `--help` returns detailed information about the `option` component of an `okv` command with the specified *`category`*, *`resource`*, and *`action`*.

Syntax: `okv` *`category resource action`* `--help`

Example Input: `okv admin endpoint provision --help`

Example Output:

```
Oracle Key Vault REST CLI Version 21.12.0.0.0 Built 10/13/2025 12:34

Usage: okv admin endpoint provision [<rest-cli-parameters>] <parameters>

The okv admin endpoint provision command downloads and installs the endpoint
software for an endpoint in the specified directory.

Command:
  <category> : admin
  <resource> : endpoint
  <action>   : provision

Required Parameters:
```

```
    --endpoint <arg>   Name of the endpoint.
    --location <arg>   Path to the location where to install the endpoint
                       software. For Transparent Data Encryption (TDE)
                       environments, specify WALLET_ROOT/okv as the
                       installation directory.

Optional Parameters:
    --auto-login <arg>   Enter one of the following values:
                         TRUE: to enable auto-login authentication
                         FALSE: (default) to store the endpoint
                         credentials that are used to connect to the
                         Oracle Key Vault server in a password-protected
                         wallet. When --auto-login is set to FALSE, then
                         you will be promoted to enter a password
                         interactively.
          --arch/arch    This option can have the value db26ai to
                         install the endpoint software to manage the
                         Oracle 23ai databases.
```

Running Oracle Key Vault RESTful Services Utility Commands

Oracle Key Vault provides a variety of ways to run RESTful services utility commands.

- RESTful Services Utility Command Syntax
  The RESTful services utility command syntax operates using the `okv` command.
- Ways of Running RESTful Services Utility Commands
  You can run the Oracle Key Vault RESTful services utility commands either by directly specifying command-specific parameters in the command line, or by using the JSON syntax.
- Creating a Script to Automatically Enroll Oracle Databases as Endpoints
  You can create a script that database administrators can run to automatically enroll Oracle Database endpoints in Oracle Key Vault.

RESTful Services Utility Command Syntax

The RESTful services utility command syntax operates using the `okv` command.

The syntax used for RESTful services utility commands is as follows:

`okv` *`category resource action rest-cli-configuration-parameters command-parameters`*

In this specification:

- *`category`* refers to the type of command you are executing, such as `managed-object`, `admin`, `cluster`, or `backup` commands.
- *`resource`* is a type of resource on which you are executing the command, such as `endpoint`, `wallet`, or `certificate`.
- *`action`* is the action to perform on the resource, such as `create`, `add`, `locate`, or `delete`.

- *rest-cli-configuration-parameters* include parameters such as `--user`, `--client_wallet`, and so on, that you specify in the REST CLI configuration file. These parameters apply to all commands.
- *command-parameters* are parameters that a command may need, such as the `--description` or `--email` parameters when you create an endpoint

In this guide, commands are identified using `okv` followed by *category*, *resource*, *action*, and if the command requires them, *rest-cli-configuration-parameters command-parameters*. For example, to create an endpoint, you would use the `okv admin endpoint create` command. This command's full syntax is as follows:

```
okv admin endpoint create --endpoint endpoint_name --description
"description" --email email_address --platform platform --type type --unique
TRUE|FALSE
```

The Oracle Key Vault RESTful services utility commands syntax follows these rules:

- It requires that you specify the command in this order: *okv category resource action rest-cli-configuration-parameters command-parameters*. You must specify the *category*, *resource*, and *action* in the order shown here. REST CLI configuration parameters must be specified before any command-specific parameters.
- It enables the configuration file (`okvrestcli.ini`) to be identified by using the `OKV_RESTCLI_CONFIG` environment variable. You set this variable in the RESTful services command line utility script `okv` itself. This frees you of the necessity of having to specify this configuration file every time that you run the command.

> **Note**
>
> For backward compatibility, the RESTful services utility command line interface that existed before Oracle Key Vault release 21.1 is still supported. You can download `okvrestclipackage.zip` to use that interface.

Most of the RESTful services utility commands support JSON input. In this guide, the commands that support JSON provide an example of how to use JSON.

Ways of Running RESTful Services Utility Commands

You can run the Oracle Key Vault RESTful services utility commands either by directly specifying command-specific parameters in the command line, or by using the JSON syntax.

- Running RESTful Services Utility Commands Using the Command Line
  You run the RESTful services utility commands from the command line by specifying all command-specific parameters in the command line.
- Running RESTful Services Utility Commands Using the JSON Syntax
  The RESTful services utility commands support JSON syntax, and after you have generated the JSON output, you can use it in combination with a command line execution of the command.
- Specifying the RESTful Services Utility Commands Output Format
  You can use the REST CLI parameter --output_format to choose the output format between JSON and text.

- Naming Conventions for Parameters Specified at the Command Line and in JSON Files
  The command parameters when specified on the command line use a different naming convention from the naming convention that is used in the JSON syntax.

Running RESTful Services Utility Commands Using the Command Line

You run the RESTful services utility commands from the command line by specifying all command-specific parameters in the command line.

For example, `okv manage-access endpoint-group add-endpoint` has the endpoint-group and endpoint parameters:

```
okv manage-access endpoint-group add-endpoint --endpoint-group
endpoint_group_name --endpoint endpoint_member
```

When specifying REST CLI configuration parameters in the command line, you must specify REST CLI configuration parameters before any command-specific parameters. In the following example, `--profile hr` is one of the *`rest_cli_configuration_parameters`*, and it is followed by the *`command_parameters`* for the `okv managed-object key create` command.

```
okv managed-object key create --profile hr --algorithm AES --length 128 --
mask "ENCRYPT,DECRYPT,EXPORT"
```

Related Topics

- Naming Conventions for Parameters Specified at the Command Line and in JSON Files
  The command parameters when specified on the command line use a different naming convention from the naming convention that is used in the JSON syntax.

Running RESTful Services Utility Commands Using the JSON Syntax

The RESTful services utility commands support JSON syntax, and after you have generated the JSON output, you can use it in combination with a command line execution of the command.

To run the RESTful services command using JSON input, you must first prepare a JSON input file with the command-specific parameter values and then run the command using parameter `--from-json` *`json-input-file`*`.json`.

The recommended process of running RESTful services utility commands using JSON input is as follows:

1. Generate JSON input designed specifically for the command, by running the command with the `--generate-json-input` parameter. For example:

   ```
   okv managed-object key create --generate-json-input
   ```

   The generated JSON input for this command is as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "key",
       "action" : "create",
       "options" : {
   ```

```
      "algorithm" : "#3DES|AES",
      "length" : "#112,168(3DES)|128,192,256(AES)",
      "mask" : [ "#ENCRYPT", "#DECRYPT", "#SIGN", "#VERIFY", "#WRAP_KEY",
"#UNWRAP_KEY", "#EXPORT", "#DERIVE_KEY", "#GENERATE_CRYPTOGRAM",
"#VALIDATE_CRYPTOGRAM", "#TRANSLATE_ENCRYPT", "#TRANSLATE_DECRYPT",
"#TRANSLATE_WRAP", "#TRANSLATE_UNWRAP" ],
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "extractable" : "#TRUE|FALSE"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ],
      "cryptoParameters" : [ {
        "blockCipherMode" : "#VALUE",
        "paddingMethod" : "#VALUE",
        "hashingAlgorithm" : "#VALUE",
        "keyRoleType" : "#VALUE"
      } ]
    }
  }
}
```

2. Save generated input to a file and then edit it so that you can perform the task. You must modify the values that begin with #. For this example, you could call the file `create_key.json` and then edit it to use the following values:

```
{
  "service": {
    "category": "managed-object",
    "resource": "key",
    "action": "create",
    "options": {
      "algorithm": "AES",
      "length": "256",
      "mask": [
        "ENCRYPT",
        "DECRYPT"
      ],
      "wallet": "hr_wallet",
      "attributes": {
         "extractable" : "FALSE"
      }
    }
  }
}
```

3. To perform the action, run the `okv managed-object key create` command with the `--from-json` parameter to specify the name of the JSON input file that you just edited.

   For example, to run the `okv managed-object key create` command by using the default configuration settings:

   ```
   okv managed-object key create --from-json create_key.json
   ```

   When using JSON input, you can also specify command parameters in the command line. The command parameters specified in the command line have higher precedence over the same parameters specified in the JSON input file.

   - **Example 1:** To create a key but with a different length than what is specified in the JSON file `create_key.json`, specify the `length` parameter in the command line:

     ```
     okv managed-object key create --from-json create_key.json --length 128
     ```

     Overriding command parameters in the command line allows use of the same JSON file for running the same command but with different parameters without having to modify the JSON input file.

   - **Example 2:** To apply the same attribute values for multiple managed objects, you specify the attribute settings in the input JSON file and specify the UUID of the object in the command line. Consider the following JSON input file `add_attributes.json`:

     ```
     {
       "service" : {
         "category" : "managed-object",
         "resource" : "attribute",
         "action" : "add",
         "options" : {
           "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
           "attributes" : {
             "contactInfo" : "psmith@example.com",
             "deactivationDate" : "2024-12-31 09:00:00",
             "name" : {
               "value" : "prod-hrdb-mkey",
               "type" : "text"
              },
             "protectStopDate" : "2024-09-30 09:00:00"
           }
         }
       }
     }
     ```

     To apply this attribute to an object with UUID `2359E04F-DA61-4F7C-BF9F-913D3369A93A`, you run:

     ```
     okv managed-object attribute add --from-json add_attributes.json --uuid
     2359E04F-DA61-4F7C-BF9F-913D3369A93A
     ```

Related Topics

- Naming Conventions for Parameters Specified at the Command Line and in JSON Files
  The command parameters when specified on the command line use a different naming convention from the naming convention that is used in the JSON syntax.

Specifying the RESTful Services Utility Commands Output Format

You can use the REST CLI parameter --output_format to choose the output format between JSON and text.

By default, the output of RESTful services utility commands is in the JSON format. However, for certain commands, you can specify the `--output_format text` to produce the command output in the text format.

For example: Below command returns the UUID of the newly created key as the text output:

```
okv managed-object key create --output_format text --algorithm AES --length
25609285F83-CC1F-4FAF-BF6C-E2262733F369
```

> **Note**
>
> The `--output_format text` is only supported for these commands:

- `okv managed-object certificate get`
- `okv managed-object certificate register`
- `okv managed-object certificate-request get`
- `okv managed-object certificate-request register`
- `okv managed-object key create`
- `okv managed-object key get`
- `okv managed-object key register`
- `okv managed-object object activate`
- `okv managed-object object destroy`
- `okv managed-object object locate`
- `okv managed-object object revoke`
- `okv managed-object opaque get`
- `okv managed-object private-key register`
- `okv managed-object public-key get`
- `okv managed-object public-key register`
- `okv managed-object secret get`
- `okv managed-object secret register`
- `okv managed-object wallet add-member`
- `okv managed-object wallet delete-member`
- `okv managed-object wallet list`

Naming Conventions for Parameters Specified at the Command Line and in JSON Files

The command parameters when specified on the command line use a different naming convention from the naming convention that is used in the JSON syntax.

The parameters in the JSON syntax use the camelCase naming convention (for example, `walletUser`, `clientWallet`). The naming convention for the parameters in the command line use follows these rules in general:

- The parameter name is prefixed by two hyphens (for example, `--user`)
- Each word is separated by a hyphen (for example, `--endpoint-group`)
- All words are in lowercase (for example, `--endpoint`)

The corresponding command line parameter names for the parameters `walletUser` and `clientWallet` from the JSON syntax are `--wallet-user` and `--client-wallet`, respectively.

Creating a Script to Automatically Enroll Oracle Databases as Endpoints

You can create a script that database administrators can run to automatically enroll Oracle Database endpoints in Oracle Key Vault.

An Oracle Key Vault administrator can create a set of scripts and files that database administrators can later download from a shared location, and run on their database servers to automatically on-board their databases into Oracle Key Vault, without any further intervention by the Oracle Key Vault administrators. As the Oracle Key Vault administrator, you will package the following:

- Oracle Key Vault RESTful services package
- `ewallet.p12` and `cwallet.sso` wallet files
- `run-me.sh` script

The following procedure explains how to create these components.

1. Download the RESTful services package and store it in your working directory, where you will also create the other files.

   ```
   curl -O -k https://Oracle_Key_Vault_IP_address:5695/okvrestclipackage.zip
   ```

2. If you have not done so already, then create a user and grant the Create Endpoint privilege to it.

   Use the Oracle Key Vault management console to create this user. For the procedure in this topic, this user will be named `restuser_ron` and will have the Create Endpoint privilege. A user with the System Administrator role creates the `restuser_ron` account and then grants the user the Create Endpoint privilege. Finally, the `restuser_ron` user must log in and change the one-time password to a permanent password. If you are preparing your Oracle Key Vault cluster to on-board ADB-on-ExaCC, additionally, the key administrator needs to grant the Create Endpoint Group privilege to `restuser_Ron` .

3. Unzip the downloaded `okvrestclipackage.zip` file into a directory where you will create the other files.

After you unzip the `okvrestclipackage.zip` file, you can use the `tree` command to see the contents of the unzipped directory structure.

```
$ tree
bin
-- okv
-- okv.bat
lib
-- okvrestcli.jar
conf
-- okvrestcli.ini
-- okvrestcli_logging.properties
```

4. Edit the `bin/okv` file.

   For example:

```
#!/bin/bash
export OKV_RESTCLI_DIR=$(dirname "${0}")/..
#export OKV_RESTCLI_CONFIG=$OKV_RESTCLI_DIR/conf/okvrestcli.ini
if [ -z "$JAVA_HOME" ]
then
  echo "JAVA_HOME environment variable is not set."
  exit 1
fi

if [ -z "$OKV_RESTCLI_CONFIG" ]
then
  echo "OKV_RESTCLI_CONFIG environment variable is not set."
  exit 1
fi

export OKV_RESTCLI_JAR=$OKV_RESTCLI_DIR/lib/okvrestcli.jar
$JAVA_HOME/bin/java -jar $OKV_RESTCLI_JAR "$@"
```

   In this specification:

   - Uncomment the line `export OKV_RESTCLI_CONFIG=$OKV_RESTCLI_DIR/conf/okvrestcli.ini`.

5. Edit the `conf/okvrestcli.ini` file.

   For example:

```
[Default]
log_property=/usr/local/okv/conf/okvrestcli_logging.properties
server=192.0.2.181
okv_client_config=/usr/local/okv/conf/okvclient.ora
user=restuser_ron
client_wallet=/home/oracle
```

   In this specification:

   - `server` is the IP address of the Oracle Key Vault server (for example, `192.0.2.181`).
   - `user` is the is the user name of the Oracle Key Vault user that you created in Step 2.

- `client_wallet` is an absolute path to a wallet that will contain the permanent password of the `restuser_ron` user. Because you are including the `user` option, the command will pick up the user's credentials from the wallet to establish a connection with the Oracle Key Vault server.

6. Run the following command, which creates a wallet and inserts the password of the `restuser_ron` user into it.

```
okv admin client-wallet add --client-wallet /home/oracle --wallet-user
restuser_ron
Password: restuser_ron_password
```

This command creates the password-protected wallet `ewallet.p12` and the auto-login wallet `cwallet.sso` in the `/home/oracle` directory.

7. Create a script similar to the `run-me.sh` script, which is part of the package that an Oracle Key Vault administrator creates for the database administrators to download.

The `run-me.sh` creates the shell script `okv-ep.sh`, which contains unique names for the virtual wallet and the associated endpoints. Use the naming convention that your site normally uses for names of wallets and other components.

```
$ more run-me.sh
#!/bin/bash
export EP_NAME=${ORACLE_SID^^}_on_${HOSTNAME/.*}
export WALLET_NAME=${ORACLE_SID^^}
curl -Ok https://192.0.2.181:5695/okvrestclipackage.zip
unzip -Vj okvrestclipackage.zip lib/okvrestcli.jar -d ./lib
cat > /home/oracle/okv-ep.sh  << EOF
#!/bin/bash
mkdir -pv ${ORACLE_BASE}/product/okv
okv manage-access wallet create --wallet ${WALLET_NAME} --unique FALSE
okv admin endpoint create --endpoint ${EP_NAME} --description "$HOSTNAME, $
(hostname -i)" --type ORACLE_DB --platform
 LINUX64 --subgroup "USE CREATOR SUBGROUP" --unique FALSE --strict-ip-
check TRUE
okv manage-access wallet set-default --wallet ${WALLET_NAME} --endpoint $
{EP_NAME}
expect << _EOF
    set timeout 120
    spawn okv admin endpoint provision --endpoint ${EP_NAME} --location $
{ORACLE_BASE}/product/okv --auto-login FALSE
    expect "Enter Oracle Key Vault endpoint password: "
    send "change-on-install\r"
    expect eof
_EOF
EOF
chmod +x okv-ep.sh
more ./okv-ep.sh
```

In this specification:

a. `export ...` creates the endpoint name from `uppercase(ORACLE_SID)_on_`*`short_hostname`* and a `WALLET_NAME` from uppercase (`ORACLE_SID`). In an Oracle Real Application Clusters (Oracle RAC) environment, replace `ORACLE_SID` with an uppercase (`ORACLE_UNQNAME`).

b. `curl ...` downloads the correct, current version of the Oracle Key Vault RESTful services package when the database administrator executes the `run-me` script.

c. `unzip ...` extracts only the `okvrestcli.jar` file and places it into the `./lib` directory.

d. `mkdir -pv ${ORACLE_BASE}/product/okv` creates the installation directory for Oracle Key Vault client software. For Oracle Database release 18c and later, this is equal to `WALLET_ROOT/okv`. (`/product/okv` is an example directory.)

e. `okv manage-access wallet create` creates a (shared) virtual wallet in Oracle Key Vault. Here, the wallet name equals `uppercase($ORACLE_SID)`.

f. `okv admin endpoint create` creates an endpoint, named after the endpoint created in the `export` command in Step a, with `type=ORACLE_DB, platform=LINUX64`, free text *`fully_qualified_hostname, IP address`*. The `--subgroup` option determines the preferred Oracle Key Vault read/write pair that the database endpoint will connect to first. Here, it is the Oracle Key Vault subgroup where the endpoint will be enrolled.

g. `okv manage-access wallet set-default` sets the default wallet, associating the endpoint created in Step f with the shared wallet created in Step e.

h. `expect` executes the `okv admin endpoint provision` command and automatically inserts a password when prompted. The benefit of using `expect` is that the password cannot be retrieved using the `ps` command.

8. Duplicate the `run-me.sh` script so that you will have a primary script and a secondary script, to be used for different situations.

   The primary script will be used for single-instance databases and the first Oracle RAC instance. The secondary script will be used for the remaining Oracle RAC nodes of a primary database and all nodes of the corresponding standby Oracle RAC database. This secondary script will associate the endpoints with the shared wallet that was created on the first instance.

   a. Rename the `run-me.sh` script to `primary-run-me.sh`.

   b. Copy `primary-run-me.sh` to a new file named `secondary-run-me.sh`.

   c. Open `secondary-run-me.sh` and remove the following line:

   ```
   okv manage-access wallet create --wallet ${ORACLE_SID^^} --unique FALSE
   ```

9. Make the scripts executable.

   ```
   $ chmod +x primary-run-me.sh
   $ chmod +x secondary-run-me.sh
   ```

10. Test each of the scripts to ensure that they can create a `okv-ep.sh` file.

    ```
    $ ./primary-run-me.sh
    $ ./secondary-run-me.sh
    ```

11. Confirm that the names for the virtual wallets and endpoints follow your naming convention by executing the following command:

    ```
    $ more okv-ep.sh
    ```

12. Create two `.zip` file packages for each of the scripts.

    Each package must have the following contents:

- `primary.zip` contains `primary-run-me.sh`, `ewallet.p12`, `cwallet.sso`, as well as the `bin` and `conf` directories. Do not include the `./lib` directory. The `./lib` library will be created and populated on demand when the `primary-run-me.sh` script is executed.
- `secondary.zip` contains `secondary-run-me.sh`, `ewallet.p12`, `cwallet.sso`, as well as the `./bin` and `./conf` directories. Do not include the `./lib` directory. The `./lib` library will be created and populated on demand when the `secondary-run-me.sh` script is executed.

13. Make these two `.zip` files available to the database administrators for them to download from a shared file server.
14. Instruct the database administrators where to download and run the scripts:
    - Run the `primary-run-me.sh` script on single-instance databases or the first Oracle RAC instance. For an Oracle Data Guard environment, run the script on the lead node of the primary Oracle RAC database.
    - Run the `secondary-run-me.sh` script on all the remaining Oracle RAC nodes of a primary database and all nodes of the corresponding standby Oracle RAC database.

Related Topics

- Step 4: Download the RESTful Services Utility
  The RESTful services utility is in the `okvrestclipackage.zip` file.

Naming Guidelines for Objects

The naming guidelines affect the following Oracle Key Vault objects: users, user groups, endpoints, endpoint groups, and virtual wallets.

The naming conventions for these objects are as follows:

- You can include the following characters in the names of endpoints, endpoint groups, user groups, and virtual wallets: letters (`a`–`z`, `A`–`Z`), numbers (`0`-`9`), underscores (`_`), periods (`.`), and hyphens (`-`).
- You can include the following characters in the names of users: letters (`a`–`z`, `A`–`Z`), numbers (`0`-`9`), and underscores (`_`).
- In most environments, the maximum number of bytes allowed for the name length is 120 bytes.
- The names of users, user groups, endpoints, and endpoint groups are not case sensitive. For example, `psmith` and `PSMITH` are considered the same user in Oracle Key Vault.
- The names of virtual wallets are case sensitive. For example, `wallet_hr` and `WALLET_HR` are considered two separate wallets in Oracle Key Vault.

How to Set the Date and Time in RESTful Services Utility Commands

You specify the date or timestamp, and duration using the supported formats.

Duration Format

The *duration* format specifies a time period or a time interval. This format is based on a subset of the ISO-8601 standard. The syntax is as follows:

```
PnYnMnDTnHnMnS
```

or

```
PnW
```

In this specification:

- `P` is the duration designator (for period) placed at the start of the duration representation.
- `n` is a numeric value.
- The capital letter following the `Pn` is a date or time value. Date value is as follows:
  - `Y` means the number of calendar years.
  - `M` means the number of calendar months.
  - `D` means the number of calendar days.
  - `W` means the number of weeks. This option cannot be specified with any other option including time values.
- `T` indicates that the remaining values represent time values, as follows:
  - `nH` means `n` hours.
  - `nM` means `n` minutes.
  - `nS` means `n` seconds.

Examples:

- 10 minutes: `PT10M`
- 2 hours 30 minutes: `PT2H30M`
- 5 hours: `PT5H`
- 3 days: `P3D`
- 45 seconds: `PT45S`
- 5 hours, 10 minutes: `PT5H10M`
- 4 weeks: `P4W`

Date and Time Formats

The date and time formats that are used in the RESTful services utility are in UTC. The four ways of setting the date and time are as follows:

- *timestamp*. This format is compatible with ISO 8601. This example translates to "start the activation date at 9 a.m. on December 31, 2024":

  ```
  "activationDate" : "2024-12-31 09:00:00",
  ```

- NOW. The following example sets the activation date to the current time when the command is run:

  ```
  "activationDate" : "NOW",
  ```

- *timestamp+duration*. This example translates to "start the activation date at 1 p.m. on December 31, 2024":

  ```
  "activationDate" : "2024-12-31 09:00:00+PT4H",
  ```

- NOW+*duration*. The following example translates to "set the activate date 10 minutes from now":

  ```
  "activationDate" : "NOW+PT10M",
  ```

> **Note**
>
> Setting the date-time attributes (such as `ActivationDate`, `DeactivationDate`, `ProcessStartDate`, and `ProtectStopDate`) for a security object to the epoch time has the same effect of not setting the attribute at all. For example, if you set the `ActivationDate` attribute of a security object to `1970-01-01 00:00:00` to immediately activate the object, then the object remains in the Pre-Active state. This is because Oracle Key Vault treats the epoch value of the attribute as if the attribute is not set at all. Listing the attributes of an object does not include date-time attributes that are set to epoch time value.

Using RESTful Services with LDAP Users

Both regular Oracle Key Vault administrators and properly authorized LDAP users can log in to a server to run Oracle Key Vault RESTful services utility commands.

When an LDAP user runs the Oracle Key Vault RESTful services utility commands, Oracle Key Vault first authenticates the user before command is run. The user's authorization that is effective for the session is determined during authentication process.

- When executing a RESTful service command, provide the user name and domain name of the user with the `--user` parameter using the following methods:
  - The LDAP user name in any of the supported formats (shown below) and the domain name separate by a vertical-bar (`|`).
    - *sAMAccountName*`|`*LDAP_domain_name*
      Example: `psmith|hr.example.com`
    - *NetBiosDomainName*`\\`*sAMAccountName*`|`*LDAP_domain_name*.
      Example: `hr\\psmith|hr.example.com`

      The double backslash (`\\`) interprets `hr\\psmith` as `hr\psmith`.
    - *userPrincipalName*`|`*LDAP_domain_name*

Example: `psmith@hr.example.com|hr.example.com`

- The user principal name of the LDAP user.
  Example: `psmith@hr.example.com`

Related Topics

- Required Privileges for Using RESTful Services
  The required RESTful services privileges are consistent with the privileges required to perform the same task in the Oracle Key Vault management console.

3

Administration Commands

You can use the administration commands to manage client wallets and endpoints.

- Client Wallet Management Commands
  You can use the client wallet management commands to manage client wallets that store user credentials. The client wallet management commands support LDAP and database users only.
- Endpoint Management Commands
  The endpoint management commands enable you to perform endpoint-related tasks such as creating or provisioning endpoints.

Client Wallet Management Commands

You can use the client wallet management commands to manage client wallets that store user credentials. The client wallet management commands support LDAP and database users only.

- okv admin client-wallet add
  The `okv admin client-wallet add` command adds the user's credentials to the client wallet. Oracle Key Vault creates the client wallets `ewallet.p12` and `cwallet.sso` if they do not exist.
- okv admin client-wallet delete
  The `okv admin client-wallet delete` command deletes a user's credentials from a client wallet.
- okv admin client-wallet list
  The `okv admin client-wallet list` command lists the users whose credentials are stored in the client wallet.
- okv admin client-wallet update
  The `okv admin client-wallet update` command updates the user's password in the client wallet.

okv admin client-wallet add

The `okv admin client-wallet add` command adds the user's credentials to the client wallet. Oracle Key Vault creates the client wallets `ewallet.p12` and `cwallet.sso` if they do not exist.

Required Authorization

None

Syntax

```
okv admin client-wallet add --client-wallet client_wallet_location --wallet-user
user_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "client-wallet",
    "action" : "add",
    "options" : {
      "clientWallet" : "#VALUE",
      "walletUser" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--client-wallet` / `clientWallet`	Required	Location of the client wallet (that is, the directory where client wallet is created)
`--wallet-user` / `walletUser`	Required	User name

JSON Example

1. Generate JSON input for the `okv admin client-wallet add` command:

   ```
   okv admin client-wallet add --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `client_wallet_add.json`) and then edit it so that you can specify the client wallet location and the user whose password you want to add to the wallet:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "client-wallet",
       "action" : "add",
       "options" : {
         "clientWallet" : "/home/oracle/okv_client_wallet",
         "walletUser" : "psmith"
       }
     }
   }
   ```

3. Run the `okv admin client-wallet add` command using the generated JSON file:

   ```
   okv admin client-wallet add --from-json client_wallet_add.json
   ```

   When prompted, enter the password for the user. After you enter the password, output similar to the following appears:

   ```
   Password: password
   {
     "result" : "Success"
   }
   ```

> **Note**
>
> Prior to Oracle Key Vault release 21.7, when the client wallet file does not have write permission to the user, 'okv admin client-wallet add' gives the success message even though it did not add the user to the wallet client file. Starting with Oracle Key Vault release 21.7, RESTful Services Utility commands reports the permission issue with an error message

okv admin client-wallet delete

The `okv admin client-wallet delete` command deletes a user's credentials from a client wallet.

Required Authorization

Read/write permissions on the client wallet

Syntax

```
okv admin client-wallet delete --client-wallet <client_wallet_location> --wallet-user
<wallet_user_name>
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "client-wallet",
    "action" : "delete",
    "options" : {
      "clientWallet" : "#VALUE",
      "walletUser" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--client-wallet` / `clientWallet`	Required	Location of the client wallet (that is, the directory where client wallet is created)
`--wallet-user` / `walletUser`	Required	User name

JSON Example

1. Generate JSON input for the `okv admin client-wallet delete` command:

   ```
   okv admin client-wallet delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `client_wallet_delete.json`) and then edit it so that you can specify the client wallet location, and the name of the user you want to remove from the wallet:

```
{
  "service" : {
    "category" : "admin",
    "resource" : "client-wallet",
    "action" : "delete",
    "options" : {
      "clientWallet" : "/home/oracle/okv_client_wallet",
      "walletUser" : "psmith"
    }
  }
}
```

3. Run the `okv admin client-wallet delete` command using the generated JSON file:

   ```
   okv admin client-wallet delete --from-json client_wallet_delete.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv admin client-wallet list

The `okv admin client-wallet list` command lists the users whose credentials are stored in the client wallet.

Required Authorization

Read file permissions on the client wallet

Syntax

```
okv admin client-wallet list --client-wallet client_wallet_location
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "client-wallet",
    "action" : "list",
    "options" : {
      "clientWallet" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--client-wallet` / `clientWallet`	Required	Location of the client wallet (that is, the directory where client wallet is created)

JSON Example

1. Generate JSON input for the `okv admin client-wallet list` command:

   ```
   okv admin client-wallet list --generate-json-input
   ```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `client_wallet_list.json`) and then modify it to include the client wallet location:

```
{
  "service" : {
    "category" : "admin",
    "resource" : "client-wallet",
    "action" : "list",
    "options" : {
      "clientWallet" : "/home/oracle/okv_client_wallet"
    }
  }
}
```

3. Run the `okv admin client-wallet list` command using the generated JSON file:

```
okv admin client-wallet list --from-json client_wallet_list.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "walletUsers" : [ "psmith", "djohn" ]
  }
}
```

okv admin client-wallet update

The `okv admin client-wallet update` command updates the user's password in the client wallet.

Required Authorization

Read/write file permissions on the wallet

Syntax

`okv admin client-wallet update --client-wallet` *`client_wallet_location`* `--wallet-user` *`user_name`*

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "client-wallet",
    "action" : "update",
    "options" : {
      "clientWallet" : "#VALUE",
      "walletUser" : "#VALUE"
    }
  }
}
```

Parameters

Parameter	Required?	Description
`--client-wallet` / `clientWallet`	Required	Location of the client wallet (that is, the directory where client wallet is created)
`--wallet-user` / `walletUser`	Required	User name

JSON Example

1. Generate JSON input for the `okv admin client-wallet update` command:

   ```
   okv admin client-wallet update --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `client_wallet_update.json`) and then edit it so that you can specify the client wallet location, and the user whose password you want to update:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "client-wallet",
       "action" : "update",
       "options" : {
         "clientWallet" : "/home/oracle/okv_client_wallet",
         "walletUser" : "psmith"
       }
     }
   }
   ```

3. Run the `okv admin client-wallet update` command using the generated JSON file:

   ```
   okv admin client-wallet update --from-json client_wallet_update.json
   ```

   When prompted, enter the password for the user. After you enter the password, output similar to the following appears:

   ```
   Password: password
   {
     "result" : "Success"
   }
   ```

Related Topics

- okv admin client-wallet list
  The `okv admin client-wallet list` command lists the users whose credentials are stored in the client wallet.

Endpoint Management Commands

The endpoint management commands enable you to perform endpoint-related tasks such as creating or provisioning endpoints.

- okv admin endpoint check-status
  The `okv admin endpoint check-status` command displays the current state of an endpoint. The state will be either `ACTIVE` or `PENDING`.

- okv admin endpoint create
  The `okv admin endpoint create` command adds a new endpoint to Oracle Key Vault.
- okv admin endpoint delete
  The `okv admin endpoint delete` command removes an endpoint from Oracle Key Vault.
- okv admin endpoint download
  The `okv admin endpoint download` command downloads the endpoint software (`okvclient.jar`) to the specified directory.
- okv admin endpoint get
  The `okv admin endpoint get` command retrieves detailed information for an endpoint, such as its endpoint group and associated wallets.
- okv admin endpoint get-enrollment-token
  The `okv admin endpoint get-enrollment-token` command retrieves an enrollment token for a registered endpoint.
- okv admin endpoint list
  The `okv admin endpoint list` command lists the endpoints along with their associated information such as creation time, certificate expiry, and default wallet.
- okv admin endpoint list-objects
  The `okv admin endpoint list-objects` command lists security objects that are associated with a specified endpoint.
- okv admin endpoint provision
  The `okv admin endpoint provision` command downloads and installs the endpoint software in the specified directory.
- okv admin endpoint re-enroll
  The `okv admin endpoint re-enroll` command re-enrolls a previously enrolled endpoint.
- okv admin endpoint re-enroll-all
  The `okv admin endpoint re-enroll-all` command re-enrolls all previously enrolled endpoints.
- okv admin endpoint resume
  The `okv admin endpoint resume` command resumes a suspended endpoint.
- okv admin endpoint suspend
  The `okv admin endpoint suspend` command suspends an endpoint.
- okv admin endpoint update
  The `okv admin endpoint update` command updates the settings of an endpoint.

okv admin endpoint check-status

The `okv admin endpoint check-status` command displays the current state of an endpoint. The state will be either `ACTIVE` or `PENDING`.

This command is meant primarily for multi-master cluster environments. However, it is still valid for other deployments and can be used to check the existence of an endpoint.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint check-status --endpoint endpoint_name|--locator-id UUID
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "check-status",
    "options" : {
      "endpoint" : "#VALUE",
      "locatorID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint` or `--locator-id` / `locatorID`	Optional	The name of the endpoint or the locator ID (universally unique ID (UUID)) of the endpoint that you want to check. The `--locator-id` / `locatorID` is required only if you are using a multi-master cluster environment. You must specify either the `--endpoint` / `endpoint` value or the `--locator-id` / `locatorID` value, not both. To find existing endpoints, run the `okv admin endpoint list` command. To find the locator ID, check the output of the `okv admin endpoint create` command that was used to create this endpoint.

JSON Example

1. Generate a JSON input template for the `okv admin endpoint check-status` command:

   ```
   okv admin endpoint check-status --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `check-status_ep.json`) and then edit it to so that you can check the endpoint. Specify either the `endpoint` value or the `locatorID` value, but not both:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "check-status",
       "options" : {
         "locatorID" : "1AC9B321-6540-4F2B-809B-95FD7416999E"
       }
     }
   }
   ```

3. Run the `okv admin endpoint check-status` command using the generated JSON file:

   ```
   okv admin endpoint check-status --from-json check-status_ep.json
   ```

   Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "status" : "ACTIVE",
    "endpoint" : "HR_DB_EP"
  }
}
```

The output includes the name of the endpoint if the endpoint object is in `ACTIVE` state. The endpoint name shown here may be different from what was specified at the endpoint creation time. If the endpoints with the same name are created on multiple cluster nodes, then Oracle Key Vault performs naming conflict resolution and it renames all but one endpoints by appending `_OKV`*`node-id`* to the endpoint name. For example, if you named the endpoint `hr_db_ep`, and there is a naming conflict, then the name could be `hr_db_ep_okv01`.

On deployments other than multi-master cluster, this command returns `Success` if the endpoint exists and output does not include entries showing the endpoint name and its state.

okv admin endpoint create

The `okv admin endpoint create` command adds a new endpoint to Oracle Key Vault.

After you add the endpoint, the endpoint will be in the **Registered** state.

Required Authorization

System Administrator role or the Create Endpoint system privilege

Syntax

```
okv admin endpoint create [--description <description>] [--email <email>] --endpoint
<endpoint> [--platform <platform>]
[--ssh-server-host-name <ssh-server-host-name>] [--strict-ip-check <strict-ip-check>] [--
subgroup <subgroup>] [--type <type>]
[--unique <unique>]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "create",
    "options" : {
      "endpoint" : "#VALUE",
      "description" : "#VALUE",
      "email" : "#VALUE",
      "platform" : "#LINUX64|SOLARIS64|SOLARIS_SPARC|HP-UX|AIX|WINDOWS|LINUX_ARM64|
LINUX_ON_SYSTEM_Z",
      "type" : "#ORACLE_DB|ORACLE_NON_DB|ORACLE_ACFS|MYSQL_DB|SSH_SERVER|OTHER",
      "subgroup" : "#VALUE|NO SUBGROUP|USE CREATOR SUBGROUP",
      "strictIpCheck" : "#TRUE|FALSE",
      "unique" : "#TRUE|FALSE",
      "sshServerHostName" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	The name of the endpoint that you want to add. See Naming Guidelines for Objects. To find existing endpoints, run the `okv admin endpoint list` command.
`--description` / `description`	Optional	A user-friendly description of the endpoint. If the description contains spaces, then you must enclose it within double quotation marks.
`--email` / `email`	Optional	Email address of the endpoint administrator. Enclose this value in double quotation marks.
`--platform` / `platform`	Optional	The endpoint platform. Allowed values are: • `AIX` • `HP-UX` • `LINUX64` (Default value) • `SOLARIS64` • `LINUX_ARM64` • `LINUX_ON_SYSTEM_Z` • `SOLARIS_SPARC` • `WINDOWS`
`--type` / `type`	Optional	Type of the endpoint. Allowed values are: • `MYSQL_DB` • `ORACLE_ACFS` • `ORACLE_DB` (Default value) • `ORACLE_NON_DB` • `SSH_SERVER` • `OTHER`
`ssh-server-host-name`	Optional	The hostname or IP address of the `SSH_SERVER` type endpoint. The `ssh-server-host-name` can be used only for `SSH_SERVER` type endpoint.
`--subgroup` / `subgroup`	Optional	For multi-master cluster environments, defines the affinity that an endpoint will have to a specific Oracle Key Vault cluster subgroup. Values are as follows: • Enter the name of a multi-master cluster subgroup. To find subgroups, in the Oracle Key Vault management console, select the **Cluster** tab, then **Management** in the left navigation bar. Subgroups for the cluster are listed under Cluster Information. • `NO SUBGROUP` creates an endpoint that will have no Oracle Key Vault cluster subgroup affinity. • `USE CREATOR SUBGROUP` creates an endpoint with affinity to the Oracle Key Vault cluster subgroup to which the node belongs where the endpoint is created.

Parameter/Template Parameter	Required?	Description
`--strict-ip-check` / `strictIpCheck`	Optional	Controls whether the Oracle Key Vault server checks the incoming IP address for a given endpoint. • `TRUE` enables Oracle Key Vault to check the incoming IP address of an endpoint. If the IP address does not match with the one that was used when the client endpoint software was installed, then Oracle Key Vault does not allow the connection. • `FALSE` disables this check and allows the incoming connection for the endpoint to come from any IP address. **Note:** If no value is entered for this option, the default value is used. The default value can be `TRUE` or `FALSE` and is determined from the value configured for the global endpoint settings.
`--unique` / `unique`	Optional	In a multi-master cluster environment, creates the endpoint as a unique endpoint. In a multi-master cluster, it is possible that an endpoint with the same name could be created from two different nodes. If that happens, then endpoint names may conflict. The Oracle Key Vault conflict resolution scheme will keep one endpoint with the given name and rename other endpoints with the conflicting names to a name using this format: *`given_ep_name`*`_OKV`*`node_id`*. Valid settings are as follows: • `TRUE` appends `_OKV`*`node_id`* to the given name and thus prevent the conflict for this endpoint name. The endpoint is immediately usable. • `FALSE` (default) causes Oracle Key Vault to begin a checking process to find if the endpoint name is unique. A unique ID is returned. You can use this ID to check the status of the endpoint creation, whether it is in progress (`PENDING`) or complete (`ACTIVE`). If the status is `PENDING`, then it is not yet usable, so any actions performed on the endpoint will fail. If the status is `ACTIVE`, then the endpoint is usable. To check the status, run the `okv admin endpoint check_status` command. If the name that you provided is already used in another node, then the name for this endpoint will have `_OKV`*`xx`* appended to it. For example, if you named the endpoint `ep12`, and there is a naming conflict, the name could be `EP12_OKV01`.
`--ssh-server-host-name` / `sshServerHostName`	Optional	The hostname or IP address of the host where you want to deploy the `SSH_SERVER` endpoint. This option can only be used with the `SSH_SERVER` type endpoint..

JSON Example

1. Generate JSON input for the `okv admin endpoint create` command:

   ```
   okv admin endpoint create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `create_ep.json`) and then edit it so that you can create the endpoint:

   ```
   {
     "service" : {
       "category" : "admin",
   ```

```
    "resource" : "endpoint",
    "action" : "create",
    "options" : {
      "endpoint" : "hr_db_ep",
      "description" : "HR database endpoint",
      "email" : "psmith@example.com",
      "platform" : "LINUX64",
      "type" : "ORACLE_DB",
      "subgroup" : "USE CREATOR SUBGROUP",
      "strictIpCheck" : "TRUE",
      "unique" : "FALSE"
    }
  }
}
```

3. Run the `okv admin endpoint create` command using the generated JSON file:

   ```
   okv admin endpoint create --from-json create_ep.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "status" : "PENDING",
       "locatorID" : "1AC9B321-6540-4F2B-809B-95FD7416999E"
     }
   }
   ```

   You can use the `locatorID` from above output with the `okv admin endpoint check-status` command to display the current state of the endpoint object. If the object status is `ACTIVE`, this command also displays the object name after the conflict-name resolution.

okv admin endpoint delete

The `okv admin endpoint delete` command removes an endpoint from Oracle Key Vault.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint delete --endpoint endpoint_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "delete",
    "options" : {
      "endpoint" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv admin endpoint delete` command:

   ```
   okv admin endpoint delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `delete_ep.json`) and then edit it so that you can delete the endpoint:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "delete",
       "options" : {
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv admin endpoint delete` command using the generated JSON file:

   ```
   okv admin endpoint delete --from-json delete_ep.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv admin endpoint download

The `okv admin endpoint download` command downloads the endpoint software (`okvclient.jar`) to the specified directory.

If you want to both download and then install the endpoint software, then use the `okv admin endpoint provision` command.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint download --endpoint endpoint_name --location download_location
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "download",
    "options" : {
      "endpoint" : "#VALUE",
      "location" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/ Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command
`--location` / `location`	Required	Absolute path to the download directory for the endpoint software. For example, if you specify `/tmp`, then the endpoint software is downloaded to `/tmp/endpoint_name/okvclient.jar`.

JSON Example

1. Generate JSON input for the `okv admin endpoint download` command:

   ```
   okv admin endpoint download --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `download_ep.json`) and then edit it so that you can download the endpoint:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "download",
       "options" : {
         "endpoint" : "hr_db_ep",
         "location": "/opt/downloads/okv"
       }
     }
   }
   ```

3. Run the `okv admin endpoint download` command using the generated JSON file:

   ```
   okv admin endpoint download --from-json download_ep.json
   ```

   A successful download of the `okvclient.jar` file displays the following output:

   ```
   {
     "result" : "Success"
   }
   ```

Related Topics

- okv admin endpoint provision
  The `okv admin endpoint provision` command downloads and installs the endpoint software in the specified directory.

okv admin endpoint get

The `okv admin endpoint get` command retrieves detailed information for an endpoint, such as its endpoint group and associated wallets.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint get --endpoint endpoint_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "get",
    "options" : {
      "endpoint" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv admin endpoint get` command:

   ```
   okv admin endpoint get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_ep.json`) and then edit it to specify an endpoint:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "get",
       "options" : {
         "endpoint" : "hr_db_ep"
       }
   ```

```
  }
}
```

3. Run the `okv admin endpoint get` command using the generated JSON file:

```
okv admin endpoint get --from-json get_ep.json
```

Output similar to the following appears:


```
{
  "result" : "Success",
  "value" : {
    "administratorEmail" : "psmith@example.com",
    "certificateExpirationTime" : "2025-09-06 05:11:14",
    "creationTime" : "2024-09-06 05:10:52",
    "defaultWallet" : "",
    "description" : "HR database endpoint",
    "effectiveEndpointConfiguration" : {
      "expirePkcs11PersistentCacheOnDatabaseShutdown" : "false",
      "serverPollTimeout" : "",
      "pkcs11ConfigulationParameterRefreshInterval" : "",
      "pkcs11InMemoryCacheTimeout" : "",
      "pkcs11PersistentCacheRefreshWindow" : "",
      "pkcs11PersistentCacheTimeout" : "",
      "pkcs11TraceDirectoryPath" : ""
    },
    "effectiveEndpointSettingsForManagedObjects" : {
      "extractableAttribute" : {
        "privateKey" : "true",
        "symmetricKey" : "true"
      }
    },
    "endpoint" : "HR_DB_EP",
    "endpointConfiguration" : {
      "expirePkcs11PersistentCacheOnDatabaseShutdown" : "false",
      "serverPollTimeout" : "",
      "pkcs11ConfigulationParameterRefreshInterval" : "",
      "pkcs11InMemoryCacheTimeout" : "",
      "pkcs11PersistentCacheRefreshWindow" : "",
      "pkcs11PersistentCacheTimeout" : "",
      "pkcs11TraceDirectoryPath" : ""
    },
    "endpointSettingsForManagedObjects" : {
      "extractableAttribute" : {
        "privateKey" : "",
        "symmetricKey" : ""
      }
    },
    "ipAddress" : "100.70.72.78",
    "lastActivityTime" : "2024-09-24 02:32:51",
    "platform" : "Linux",
    "selfEnrolled" : "false",
    "status" : "Enrolled",
    "strictIpCheck" : "true",
    "strictIpCheckConfiguration" : "",
    "type" : "Oracle Database"
  }
}
```


The output under `effectiveEndpointConfiguration` and `effectiveEndpointSettingsForManagedObjects` represents the settings that are used for the endpoint configuration (hence, the use of `effectiveEndpoint...` in the name). They

are determined based on the endpoint-specific settings and the global endpoint specific settings.

okv admin endpoint get-enrollment-token

The `okv admin endpoint get-enrollment-token` command retrieves an enrollment token for a registered endpoint.

The enrollment token is a one-time token that is generated during the endpoint creation (registration). Oracle Key Vault uses this token to download the software and install the endpoint. The `okv admin endpoint get-enrollment-token` is useful for the cases where the endpoint administrator (and not the Oracle Key Vault administrator) must download and provision the endpoint. These endpoint administrators, who generally are not Oracle Key Vault users, use the Oracle Key Vault management console to download the endpoint software by providing the token. The `okv admin endpoint get-enrollment-token` command enables the Oracle Key Vault administrator to retrieve the token using the RESTful services utility, and then pass it securely to an endpoint administrator through an out-of-band channel (for example, email).

This command will work only for endpoints that are in the **Registered** state.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint get-enrollment-token --endpoint endpoint_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "get-enrollment-token",
    "options" : {
      "endpoint" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the registered endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv admin endpoint get-enrollment-token` command:

   ```
   okv admin endpoint get-enrollment-token --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_token.json`) and then edit it so that you can get the enrollment token:

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "get-enrollment-token",
    "options" : {
      "endpoint" : "hr_db_ep"
    }
  }
}
```

3. Run the `okv admin endpoint get-enrollment-token` command using the generated JSON file:

```
okv admin endpoint get-enrollment-token --from-json get_token.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "token" : "Si71duR2mGQ8naSZ"
  }
}
```

okv admin endpoint list

The `okv admin endpoint list` command lists the endpoints along with their associated information such as creation time, certificate expiry, and default wallet.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint list [--limit <limit>] [--platform <platform>]
[--status <status>] [--type <type>] [--last-active-duration <last-active-duration>] [--
last-inactive-duration <last-inactive-duration>]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "list",
    "options" : {
      "platform" : "#LINUX64|SOLARIS64|SOLARIS_SPARC|HP-UX|AIX|WINDOWS",
      "type" : "#ORACLE_DB|ORACLE_NON_DB|ORACLE_ACFS|MYSQL_DB|SSH_SERVER|OTHER",
      "status" : "#REGISTERED|ENROLLED|SUSPENDED",
      "limit" : "#VALUE",
      "lastActiveDuration" : "#PnYnMnWnDTnHnMnS",
      "lastInactiveDuration" : "#PnYnMnWnDTnHnMnS"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--limit` / `limit`	Optional	Number of endpoints to list. Enter any whole number from 1 and higher. If the limit is specified, then Oracle Key Vault fetches the number of endpoints up to the specified limit. If the limit is not specified, then Oracle Key Vault fetches up to 10,000 endpoints. If you specify a value that is greater than 10,000, then Oracle Key Vault will attempt to fetch those many endpoints, depending on the server, client, and network resources. In the output that you retrieve, the `fetchedObjectCount` value lists the actual number of endpoints that are fetched. For example, if you specify `100` as the limit but there are only 50 endpoints fetched, then Oracle Key Vault sets `fetchedObjectCount` to `50`. If you omit this parameter, then Oracle Key Vault retrieves up to 10,000 endpoints. For another example, if the limit is 100 and `fetchedObjectCount` is `100`, then this means that there are more endpoints. To fetch all endpoints, you need to run this command with an increased value for the `--limit` parameter. If `fetchedObjectCount` is less than the specified limit, then it means that you have retrieved all the available endpoints.
`--platform` / `platform`	Optional	The endpoint platform. The allowed values are: • `AIX` • `HP-UX` • `LINUX64` • `SOLARIS64` • `SOLARIS_SPARC` • `WINDOWS` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supporting an OR operation.
`--type` / `type`	Optional	Type of the endpoint platform. The allowed values are: • `MYSQL_DB` • `ORACLE_ACFS` • `ORACLE_DB` • `ORACLE_NON_DB` • `SSH_SERVER` • `OTHER` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supporting an OR operation.
`--status` / `status`	Optional	Status of the endpoint platform. The allowed values are: • `REGISTERED` • `ENROLLED` • `SUSPENDED` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supporting an OR operation.

Parameter/Template Parameter	Required?	Description
`--last-active-duration` / `lastActiveDuration`	Optional	Specifies the endpoint with last activity within the defined duration. **Note:** Enter a value for this parameter using the ISO 8601 standard. For example, enter PT10M to specify an interval size of 10 Minutes. Example: ISO 8601 Duration.
`--last-inactive-duration` / `lastInactiveDuration`	Optional	Specifies the endpoint with no activity within the defined duration. **Note:** Enter a value for this parameter using the ISO 8601 standard. For example, enter PT10M to specify an interval size of 10 Minutes. Example: ISO 8601 Duration.

JSON Example

1. Generate a JSON input template for the `okv admin endpoint list` command:

   ```
   okv admin endpoint list --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `list_ep.json`) and then edit it to specify the number of objects to fetch:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "list",
       "options" : {
         "limit" : "2"
       }
     }
   }
   ```

3. Run the `okv admin endpoint list` command using the generated JSON file:

   ```
   okv admin endpoint list --from-json list_ep.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "endpoints" : [ {
         "commonNameOfCertificateIssuer" : "CA",
         "createdBy" : "OKVADMIN",
         "creationTime" : "2024-09-17 01:04:00",
         "lastActivityTime" : "2024-10-12 10:45:00",
         "defaultWallet" : "",
         "description" : "",
         "endpoint" : "HR_DB_EP",
         "endpointCertificateExpiration" : "2025-09-17 01:04:22",
         "enrollmentToken" : "",
         "ipAddress" : "",
         "platform" : "Linux",
         "status" : "Enrolled",
         "type" : "Oracle Database"
       }, {
         "commonNameOfCertificateIssuer" : "CA",
         "createdBy" : "OKVADMIN",
   ```

```
        "creationTime" : "2024-09-06 05:10:52",
        "lastActivityTime" : "2024-09-24 02:32:51",
        "defaultWallet" : "",
        "description" : "",
        "endpoint" : "EPOINT1",
        "endpointCertificateExpiration" : "2025-09-06 05:11:14",
        "enrollmentToken" : "",
        "ipAddress" : "100.70.72.78",
        "platform" : "Linux",
        "status" : "Enrolled",
        "type" : "Oracle Database"
      } ],
      "fetchedObjectCount" : "2"
    }
  }
```

okv admin endpoint list-objects

The `okv admin endpoint list-objects` command lists security objects that are associated with a specified endpoint.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint list-objects --endpoint <endpoint> [--exclude-wallet-membership
<exclude-wallet-membership>]
    [--limit <limit>][--state <state>] [--type <type>]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "list-objects",
    "options" : {
      "endpoint" : "#VALUE",
      "state" : "#PRE-ACTIVE|ACTIVE|DEACTIVATED|COMPROMISED|DESTROYED|
DESTROYED_COMPROMISED",
      "type" : "#CERTIFICATE|OPAQUE|PRIVATE_KEY|PUBLIC_KEY|SECRET|SYMMETRIC_KEY",
      "limit" : "#VALUE",
      "excludeWalletMembership" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

Parameter/Template Parameter	Required?	Description
`--limit` / `limit`	Optional	Number of objects to list. Enter any whole number from 1 and higher. If the limit is specified, then Oracle Key Vault fetches the number of objects up to the specified limit. If the limit is not specified, then Oracle Key Vault fetches up to 10,000 objects. If you specify a value that is greater than 10,000, then Oracle Key Vault will attempt to fetch that value, depending on the server, client, and network resources. In the output that you retrieve, the `fetchedObjectCount` value lists the actual number of objects that are fetched. For example, if you specify `100` as the limit but there are only 50 objects fetched, then Oracle Key Vault sets `fetchedObjectCount` to `50`. If you omit this parameter, then Oracle Key Vault retrieves up to 10,000 objects. For another example, if the limit is 100 and `fetchedObjectCount` is `100`, then this means that there are more objects. To fetch all objects, you need to run this command with an increased value for the `--limit` parameter. If `fetchedObjectCount` is less than the specified limit, then it means that you have retrieved all the available objects.
`--exclude-wallet-membership` / `excludeWalletMembership`	Optional	Controls whether wallet membership is shown for each object. • `TRUE` disables the showing of wallet membership for endpoint objects. • `FALSE` (default) enables the showing of wallet membership for endpoint objects.
`--type` / `type`	Optional	Type of the security object. The allowed values are: • `CERTIFICATE` • `OPAQUE` • `PRIVATE_KEY` • `PUBLIC_KEY` • `SECRET` • `SYMMETRIC_KEY` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supporting an OR operation.
`--state` / `state`	Optional	State of the security object. The allowed values are: • `PRE-ACTIVE` • `ACTIVE` • `DEACTIVATED` • `COMPROMISED` • `DESTROYED` • `DESTROYED_COMPROMISED` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supporting an OR operation.

JSON Example

1. Generate a JSON input template for the `okv admin endpoint list-objects` command:

```
okv admin endpoint list-objects --generate-json-input
```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `list-obj_ep.json`) and then edit it to specify an endpoint the number of records:

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "list-objects",
    "options" : {
      "endpoint" : "sales-ep",
      "limit" : "8"
    }
  }
}
```

3. Run the `okv admin endpoint list-objects` command using the generated JSON file:

```
okv admin endpoint list-objects --from-json list-obj_ep.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "fetchedObjectCount" : "8",
    "managedObjects" : [ {
      "creatingEndpoint" : "SALES-EP",
      "creationDate" : "2021-08-04 18:34:52",
      "deactivationDate" : "2051-10-10 10:10:10",
      "displayName" : "X.509 Certificate: DN EMAILADDRESS=psmith@example.com,
CN=vienna, OU=Security, O=Oracle, L=Reston, ST=VA, C=US",
      "name" : "ps30090",
      "protectStopDate" : "2053-10-10 10:10:10",
      "state" : "PRE-ACTIVE",
      "type" : "CERTIFICATE",
      "uuid" : "847D1538-915D-4FD7-BF14-829B1A11FAF9"
    }, {
      "creatingEndpoint" : "SALES-EP",
      "creationDate" : "2021-08-03 21:40:25",
      "deactivationDate" : "2029-12-25 15:11:11",
      "displayName" : "Symmetric Key: Name Sales Key 1",
      "name" : "sales_key_1",
      "protectStopDate" : "",
      "state" : "Pre-Active",
      "type" : "Symmetric Key",
      "uuid" : "670B600E-1667-4FD1-BF94-C35C4BC81E8B"
    }, {
      "creatingEndpoint" : "SALES-EP",
      "creationDate" : "2021-08-03 21:21:33",
      "deactivationDate" : "2029-12-25 15:11:11",
      "displayName" : "X.509 Certificate: DN EMAILADDRESS=psmith@example.com,
CN=vienna, OU=Security, O=Oracle, L=Reston, ST=VA, C=US",
      "name" : "orders_key_1",
      "protectStopDate" : "2029-12-25 15:11:11",
      "state" : "Pre-Active",
      "type" : "Certificate",
      "uuid" : "0C11B125-B17A-4F90-BF16-F876E5E20A21"
    }, {
      "creatingEndpoint" : "SALES-EP",
      "creationDate" : "2021-08-03 13:36:01",
```

```
      "deactivationDate" : "",
      "displayName" : "rec_key_1",
      "protectStopDate" : "",
      "state" : "Pre-Active",
      "type" : "Symmetric Key",
      "uuid" : "780608F6-0CA6-4FC5-BF46-A7B8A36074F7"
    }, {
      "creatingEndpoint" : "SALES-EP",
      "creationDate" : "2021-08-02 15:41:38",
      "deactivationDate" : "2029-12-25 15:11:11",
      "displayName" : "X.509 Certificate: DN EMAILADDRESS=psmith@example.com,
CN=vienna, OU=Security, O=Oracle, L=Reston, ST=VA, C=US",
      "name" : "cert_key_1",
      "protectStopDate" : "2029-12-25 15:11:11",
      "state" : "Pre-Active",
      "type" : "Certificate",
      "uuid" : "72EA8183-98BA-4F5A-BF31-CE7256E29496"
    }, {
      "creatingEndpoint" : "SALES-EP",
      "creationDate" : "2021-07-26 20:19:32",
      "deactivationDate" : "2029-12-25 15:11:11",
      "displayName" : "X.509 Certificate: DN EMAILADDRESS=psmith@example.com,
CN=vienna, OU=Security, O=Oracle, L=Reston, ST=VA, C=US",
      "name" : "emp_key_1",
      "protectStopDate" : "2029-12-25 15:11:11",
      "state" : "Pre-Active",
      "type" : "Certificate",
      "uuid" : "975F17DF-11C1-4F16-BFBC-28E9C200C99F"
    }, {
      "creatingEndpoint" : "SALES-EP",
      "creationDate" : "2021-07-23 17:22:14",
      "deactivationDate" : "2041-10-10 10:10:10",
      "displayName" : "emp_key_2",
      "protectStopDate" : "",
      "state" : "Active",
      "type" : "Symmetric Key",
      "uuid" : "330F5527-0DB2-4FD1-BF54-1FA189C8A765"
    }, {
      "creatingEndpoint" : "SALES-EP",
      "creationDate" : "2021-06-30 21:01:48",
      "deactivationDate" : "",
      "displayName" : "Symmetric Key: Name psc7",
      "name" : "emp_key_2,emp_key_3,emp_key_5,emp_key_6,emp_key_7",
      "protectStopDate" : "",
      "state" : "Active",
      "type" : "Symmetric Key",
      "uuid" : "7432AED6-6628-4F43-BF7C-9D30023A4301"
    } ]
  }
}
```


okv admin endpoint provision

The `okv admin endpoint provision` command downloads and installs the endpoint software in the specified directory.

This directory should have read, write, and execute permissions for the owner and its group. For example, if the Oracle Key Vault endpoint software is installed in an Oracle Database server, then this endpoint installation directory should have read, write, and execute permissions by the `oracle` user and the `oinstall` group. This ensures that processes can access directories appropriately at run time.

You must meet the following prerequisites to run this command:

- You must be a user with System Administrator role or the Manage Endpoint object privilege for the endpoint.
- You must ensure that the soft link `/usr/bin/java` points to `$ORACLE_HOME/jdk/jre/bin/java`.
- You must know how the installation process determines the location of the `okvclient.ora` file.

If you only want to download the endpoint software but not install it, then use the `okv admin endpoint download` command.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint provision --endpoint endpoint_name --location software_location --
auto-login TRUE|FALSE --arch db26ai
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "provision",
    "options" : {
      "endpoint" : "#VALUE",
      "location" : "#VALUE",
      "arch" : "#db26ai",
      "autoLogin" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.
`--location` / `location`	Required	Path to the location where to install the endpoint software. For Transparent Data Encryption (TDE) environments, specify *`WALLET_ROOT`*`/okv` as the installation directory.
`--arch/arch`	Optional	This option takes the value `db26ai` to install the endpoint software to support Oracle AI 26ai databases running in FIPS mode using OpenSSL libraries.

Parameter/Template Parameter	Required?	Description
`--auto-login` / `autoLogin`	Optional	Enter one of the following values: • TRUE to enable auto-login authentication • FALSE (default) to store the endpoint credentials that are used to connect to the Oracle Key Vault server in a password-protected wallet. When `--auto-login` is set to FALSE, then you will be prompted to enter a password interactively.

JSON Example

1. Generate JSON input for the `okv admin endpoint provision` command:

   ```
   okv admin endpoint provision --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `provision_ep.json`) and then edit it so that you can download and install the endpoint software:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "provision",
       "options" : {
         "endpoint" : "hr_db_ep",
         "location" : "/u01/opt/oracle/product/okv",
         "autoLogin" : "TRUE"
       }
     }
   }
   ```

3. Run the `okv admin endpoint provision` command using the generated JSON file:

   ```
   okv admin endpoint provision --from-json provision_ep.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

Related Topics

- okv admin endpoint download
  The `okv admin endpoint download` command downloads the endpoint software (`okvclient.jar`) to the specified directory.
- Location of the okvclient.ora File and Environment Variables

okv admin endpoint re-enroll

The `okv admin endpoint re-enroll` command re-enrolls a previously enrolled endpoint.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint re-enroll --endpoint endpoint_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "re-enroll",
    "options" : {
      "endpoint" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv admin endpoint re-enroll` command:

   ```
   okv admin endpoint re-enroll --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generate input to a file (for example, `re-enroll_ep.json`) and then edit it so that you can re-enroll the endpoint:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "re-enroll",
       "options" : {
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv admin endpoint re-enroll` command using the generated JSON file:

   ```
   okv admin endpoint re-enroll --from-json re-enroll_ep.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv admin endpoint re-enroll-all

The `okv admin endpoint re-enroll-all` command re-enrolls all previously enrolled endpoints.

Required Authorization

System Administrator role

> **Note**
>
> Oracle recommends that you exercise caution when you run this command as it suspends all endpoints prior to re-enrolling them.

Syntax

```
okv admin endpoint re-enroll-all
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "re-enroll-all"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv admin endpoint re-enroll-all` command:

   ```
   okv admin endpoint re-enroll-all --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file (for example, `re-enroll-all_ep.json`).
3. Run the `okv admin endpoint re-enroll-all` command using the generated JSON file:

   ```
   okv admin endpoint re-enroll-all --from-json re-enroll-all_ep.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv admin endpoint resume

The `okv admin endpoint resume` command resumes a suspended endpoint.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint resume --endpoint endpoint_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "resume",
    "options" : {
      "endpoint" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the suspended endpoint. To find the names of suspended endpoints, run the `okv admin endpoint list` command, and in the output, look for the endpoints that have a status of `Suspended`.

JSON Example

1. Generate JSON input for the `okv admin endpoint resume` command:

   ```
   okv admin endpoint resume --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `resume_ep.json`) and then edit it to specify the suspended endpoint:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "resume",
       "options" : {
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv admin endpoint resume` command using the generated JSON file:

```
okv admin endpoint resume --from-json resume_ep.json
```

Output similar to the following appears:

```
{
  "result" : "Success"
}
```

okv admin endpoint suspend

The `okv admin endpoint suspend` command suspends an endpoint.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint

Syntax

```
okv admin endpoint suspend --endpoint endpoint_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "suspend",
    "options" : {
      "endpoint" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv admin endpoint suspend` command:

   ```
   okv admin endpoint suspend --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `suspend_ep.json`) and then edit it to specify the endpoint to suspend:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "endpoint",
       "action" : "suspend",
       "options" : {
         "endpoint" : "hr_db_ep"
   ```

```
    }
  }
}
```

3. Run the `okv admin endpoint suspend` command using the generated JSON file:

```
okv admin endpoint suspend --from-json suspend_ep.json
```

Output similar to the following appears:

```
{
  "result" : "Success"
}
```

okv admin endpoint update

The `okv admin endpoint update` command updates the settings of an endpoint.

Required Authorization

System Administrator role or the Manage Endpoint object privilege for the endpoint.

> **Note**
>
> To update `endpointSettingsForManagedObject`, you need to have a key admin role, also to update `endpointConfiguration`, you need to have a system administrator role or Managed endpoint privilege.

Syntax

```
okv admin endpoint update [--description <description>] [--email <email>] --endpoint
current_endpoint_name [--name <name>]
[--platform <platform>] [--ssh-server-host-name <ssh-server-host-name>] [--strict-ip-
check <strict-ip-check>] [--subgroup <subgroup>]
[--type <type>] [--unique <unique>]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "update",
    "options" : {
      "endpoint" : "#VALUE",
      "name" : "#VALUE",
      "description" : "#VALUE",
      "email" : "#VALUE",
      "platform" : "#LINUX64|SOLARIS64|SOLARIS_SPARC|HP-UX|AIX|WINDOWS|LINUX_ARM64|
LINUX_ON_SYSTEM_Z",
      "type" : "#ORACLE_DB|ORACLE_NON_DB|ORACLE_ACFS|MYSQL_DB|OTHER",
      "subgroup" : "#VALUE|NO SUBGROUP|USE CREATOR SUBGROUP",
      "unique" : "#TRUE|FALSE",
      "sshServerHostName" : "#VALUE",
      "strictIpCheck" : "#TRUE|FALSE",
      "endpointConfiguration" : {
        "expirePkcs11PersistentCacheOnDatabaseShutdown" : "#TRUE|FALSE",
        "serverPollTimeout" : "#VALUE",
        "pkcs11ConfigurationParameterRefreshInterval" : "#VALUE",
        "pkcs11InMemoryCacheTimeout" : "#VALUE",
```

```
        "pkcs11PersistentCacheRefreshWindow" : "#VALUE",
        "pkcs11PersistentCacheTimeout" : "#VALUE",
        "pkcs11TraceDirectoryPath" : "#VALUE"
      },
      "endpointSettingsForManagedObjects" : {
        "extractableAttribute" : {
          "privateKey" : "#TRUE|FALSE",
          "symmetricKey" : "#TRUE|FALSE"
        }
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint that you want to update. To find existing endpoints, run the `okv admin endpoint list` command.
`--name` / `name`	Optional	A new name for the endpoint. Ensure that you follow the naming guidelines for objects. See Naming Guidelines for Objects.
`--description` / `description`	Optional	A user-friendly description of the endpoint. If the description contains spaces, then you must enclose it within double quotation marks.
`--email` / `email`	Optional	Email address of the endpoint administrator. Enclose this value in double quotation marks.
`--platform` / `platform`	Optional	The endpoint platform. Allowed values are: • `AIX` • `HP-UX` • `LINUX64` • `LINUX_ARM64` • `LINUX_ON_SYSTEM_Z` • `SOLARIS64` • `SOLARIS_SPARC` • `WINDOWS`
`--type` / `type`	Optional	Type of the endpoint. Allowed values are: • `MYSQL_DB` • `ORACLE_ACFS` • `ORACLE_DB` • `ORACLE_NON_DB` • `OTHER`

Parameter/Template Parameter	Required?	Description
`--subgroup` / `subgroup`	Optional	For multi-master cluster environments, defines the affinity that an endpoint will have to a specific Oracle Key Vault cluster subgroup. Values are as follows: • Enter the name of a multi-master cluster subgroup. To find subgroups, in the Oracle Key Vault management console, select the **Cluster** tab, then **Management** in the left navigation bar. Subgroups for the cluster are listed under Cluster Information. • `NO SUBGROUP` creates an endpoint that will have no Oracle Key Vault cluster subgroup affinity. • `USE CREATOR SUBGROUP` creates an endpoint with affinity to the Oracle Key Vault cluster subgroup to which the node that the endpoint is created in belongs.
`--unique` / `unique`	Optional	In a multi-master cluster environment, creates the endpoint as a unique endpoint. In a multi-master cluster, it is possible that an endpoint with the same name could be created from two different nodes. If that happens, then endpoint names may conflict. The Oracle Key Vault conflict resolution scheme will keep one endpoint with the given name and rename other endpoints with the conflicting names to a name using this format: *`given_ep_name`*`_OKV`*`node_id`*. Valid settings are as follows: • `TRUE` appends `_OKV`*`node_id`* to the given name and thus prevent the conflict for this endpoint name. • `FALSE` (default) causes Oracle Key Vault to begin a checking process to find if the endpoint name is unique. A unique ID is returned. You can use this ID to check the status of the endpoint creation, whether it is in progress (`PENDING`) or complete (`ACTIVE`). If the status is `PENDING`, then it is not yet usable, so any actions performed on the endpoint will fail. If the status is `ACTIVE`, then the endpoint is usable. To check the status, run the `okv admin endpoint check_status` command. If the name that you provided is already used in another node, then the name for this endpoint will have `_OKV`*`xx`* appended to it. For example, if you named the endpoint `ep12`, and there is a naming conflict, the name could be `EP12_OKV01`.

Parameter/Template Parameter	Required?	Description
`--strict-ip-check / strictIpCheck`	Optional	Controls whether the Oracle Key Vault server checks the incoming IP address for a given endpoint. • TRUE enables Oracle Key Vault to check the incoming IP address of an endpoint. If the IP address does not match with the one that was used when the client endpoint software was installed, then Oracle Key Vault does not allow the connection. • FALSE disables this check and allows the incoming connection for the endpoint to come from any IP address. **Note** If an empty value (default) is entered for this option, the endpoint will inherit this setting from the global endpoint settings.

Parameter/Template Parameter	Required?	Description

`/endpointConfiguration`	Optional	Each `endpointConfiguration` setting represents an endpoint configuration parameter. You must use the JSON syntax to add an endpoint configuration parameter(s). You cannot specify these endpoint configuration parameter(s) at the command line.
		Values that you can enter are as follows:
		• `expirePkcs11PersistentCacheOnDatabaseShutdown` sets whether the PKCS#11 persistent cache for a given endpoint database automatically expires upon shutdown of the endpoint database.
		– `TRUE` (default) enables `expirePkcs11PersistentCacheOnDatabaseShutdown`.
		– `FALSE` disables `expirePkcs11PersistentCacheOnDatabaseShutdown`.
		For any endpoint-specific configuration parameters, empty value is not supported.
		• The following timeout parameters:
		– `serverPollTimeout` specifies the server poll timeout.
		– `pkcs11ConfigurationParameterRefreshInterval` sets the frequency at which a long-running process will re-read the `okvclient.ora` configuration file.
		– `pkcs11InMemoryCacheTimeout` sets how long a master encryption key is available in the in-memory cache.
		– `pkcs11PersistentCacheRefreshWindow` extends the time the master encryption key remains available for use in the persistent cache after its persistent cache timeout period has expired.
		– `pkcs11PersistentCacheTimeout` sets how long the master encryption is available in the persistent cache.
		You can use different ways to set these timeout values, which use the duration format based on the ISO-8601 standard. Examples are as follows:
		`"serverPollTimeout" : "PT0.3S", -- 300 millseconds`
		`"pkcs11InMemoryCacheTimeout" : "PT10M", -- 10 minutes`
		`"pkcs11PersistentCacheTimeout" : "PT5H", -- 5 hours`
		`"pkcs11PersistentCacheRefreshWindow" : "P1D", -- 1 day`
		• `pkcs11TraceDirectoryPath` sets the path of the trace files

Parameter/Template Parameter	Required?	Description
/ endpointSettingsForManagedObjects	Optional	extractableAttribute enables you to specify whether symmetric keys can be extracted from Oracle Key Vault. You must use the JSON syntax to add or modify this setting. You cannot specify this setting at the command line. The setting is as follows: • symmetricKey – TRUE allows the symmetricKey object value to be extracted from Oracle Key Vault. – FALSE prevents the symmetricKey object value from being extracted from Oracle Key Vault. • privateKey – TRUE allows the privateKey object value to be extracted from Oracle Key Vault. – FALSE prevents the privateKey object value from being extracted from Oracle Key Vault. For any endpoint-specific settings, you can also set them to an empty value (default) which signifies the absence of an endpoint specific setting. In this case, the endpoint inherits this setting from the global endpoint settings. You remove an endpoint specific-setting by specifying an empty value for it. After an endpoint-specific setting is removed, the endpoint starts inheriting the setting from the global endpoint settings. The default value for the symmetricKey and privateKey setting is inherited from the global endpoint configuration. If you provide an empty value for endpointConfiguration or extractableAttribute, then it will remove the corresponding configuration.

CLI Example

```
okv admin endpoint update [--description <description>] [--email <email>] --
endpoint <endpoint> [--name <name>] [--platform <platform>]
       [--ssh-server-host-name <ssh-server-host-name>] [--strict-ip-check
<strict-ip-check>] [--subgroup <subgroup>] [--type <type>] [--unique
       <unique>]
```

JSON Example

1. Generate JSON input for the `okv admin endpoint update` command:

   ```
   okv admin endpoint update --generate-json-input
   ```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `update_ep.json`) and then edit it to update the endpoint:

```
{
  "service": {
    "category": "admin",
    "resource": "endpoint",
    "action": "update",
    "options": {
      "endpoint": "hr_db_ep",
      "description": "",
      "platform": "LINUX64",
      "email": "",
      "type": "ORACLE_DB",
      "strictIpCheck" : "TRUE",
      "endpointConfiguration" : {
          "expirePkcs11PersistentCacheOnDatabaseShutdown" : "TRUE",
          "serverPollTimeout" : "PT5M",
          "pkcs11ConfigurationParameterRefresh" : "PT11M",
          "pkcs11InMemoryCacheTimeout" : "PT20M",
          "pkcs11PersistentCacheRefreshWindow" : "PT30M",
          "pkcs11PersistentCacheTimeout" : "PT40M",
          "pkcs11TraceDirectoryPath" : "/users/psmith/work"
      },
      "endpointSettingsForManagedObjects" : {
          "extractableAttribute" : {
              "symmetricKey" : "FALSE"
        }
      }
    }
  }
}
```

3. Run the `okv admin endpoint update` command using the generated JSON file:

```
okv admin endpoint update --from-json update_ep.json
```

Output similar to the following appears:

```
{
  "result" : "Success"
}
```

Related Topics

- Naming Guidelines for Objects
  The naming guidelines affect the following Oracle Key Vault objects: users, user groups, endpoints, endpoint groups, and virtual wallets.
- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.
- *Oracle Key Vault Administrator's Guide*

4

Access Management Commands

You can use the access management commands to manage wallets and endpoint groups.

- okv manage-access endpoint-group add-endpoint
  The `okv manage-access endpoint-group add-endpoint` command adds an existing endpoint to an endpoint group.
- okv manage-access endpoint-group check-status
  The `okv manage-access endpoint-group check-status` command checks the naming conflict resolution status of an endpoint group in a multi-master cluster.
- okv manage-access endpoint-group create
  The `okv manage-access endpoint-group create` command creates a new endpoint group.
- okv manage-access endpoint-group delete
  The `okv manage-access endpoint-group delete` command deletes an endpoint group.
- okv manage-access endpoint-group get
  The `okv manage-access endpoint-group get` command retrieves detailed information about an endpoint group, such as its member endpoints and wallet access.
- okv manage-access endpoint-group list
  The `okv manage-access endpoint-group list` command retrieves a list of endpoint groups and their associated information.
- okv manage-access endpoint-group remove-endpoint
  The `okv manage-access endpoint-group remove-endpoint` command removes an endpoint from an endpoint group.
- okv manage-access endpoint-group update
  The `okv manage-access endpoint-group update` command changes the name and description of an endpoint group, and can be used to ensure that the endpoint group name is unique.
- okv manage-access wallet add-access
  The `okv manage-access wallet add-access` command grants an endpoint or an endpoint group a level of access to a wallet.
- okv manage-access wallet add-object
  The `okv manage-access wallet add-object` command adds a security object to a wallet.
- okv manage-access wallet check-status
  The `okv manage-access wallet check-status` command checks the naming conflict resolution status of a wallet in a multi-master cluster.
- okv manage-access wallet create
  The `okv manage-access wallet create` command creates a wallet.
- okv manage-access wallet delete
  The `okv manage-access wallet delete` command deletes a wallet.
- okv manage-access wallet get
  The `okv manage-access wallet get` command retrieves information about a specified wallet, such as the default wallet name and the wallet access.

- okv manage-access wallet get-default
  The `okv manage-access wallet get-default` command gets the default wallet that has been associated with an endpoint.
- okv manage-access wallet list
  The `okv manage-access wallet list` command lists wallets on which some level of access is granted to the user.
- okv manage-access wallet list-endpoint-wallets
  The `okv manage-access wallet list-endpoint-wallets` command lists the wallets that are associated with an endpoint.
- okv manage-access wallet list-objects
  The `okv manage-access wallet list-objects` command retrieves the security objects that are members of the specified wallet.
- okv manage-access wallet list-objects-wallets
- okv manage-access wallet remove-access
  The `okv manage-access wallet remove-access` command removes the access that an endpoint or an endpoint group has to a wallet.
- okv manage-access wallet remove-object
  The `okv manage-access wallet remove-object` command removes a security object from a wallet.
- okv manage-access wallet set-default
  The `okv manage-access wallet set-default` command sets the default wallet for an endpoint.
- okv manage-access wallet update
  The `okv manage-access wallet update` command updates a wallet.
- okv manage-access wallet update-access
  The `okv manage-access wallet update-access` command updates the level of access that an endpoint or an endpoint group has to a wallet.

okv manage-access endpoint-group add-endpoint

The `okv manage-access endpoint-group add-endpoint` command adds an existing endpoint to an endpoint group.

Required Authorization

Key Administrator role or the Manage Endpoint Group object privilege for the endpoint group

Syntax

```
okv manage-access endpoint-group add-endpoint --endpoint-group endpoint_group_name --
endpoint endpoint_member
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "add-endpoint",
    "options" : {
      "endpointGroup" : "#VALUE",
      "endpoint" : "#VALUE"
    }
```

```
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint-group` / `endpointGroup`	Required	Name of the endpoint group. To find existing endpoint groups, run the `okv manage-access endpoint-group list` command.
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv manage-access endpoint-group add-endpoint` command:

   ```
   okv manage-access endpoint-group add-endpoint --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `add_ep_to_group.json`) and then edit it to add the endpoint to an endpoint group:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "endpoint-group",
       "action" : "add-endpoint",
       "options" : {
         "endpointGroup" : "epg_hr",
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv manage-access endpoint-group add-endpoint` command using the generated JSON file:

   ```
   okv manage-access endpoint-group add-endpoint --from-json add_ep_to_group.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv manage-access endpoint-group check-status

The `okv manage-access endpoint-group check-status` command checks the naming conflict resolution status of an endpoint group in a multi-master cluster.

This command is meant primarily for multi-master cluster environments. However, it is valid for other deployments and can be used to check the existence of an endpoint group.

Required Authorization

Key Administrator role or the Manage Endpoint Group object privilege for the endpoint group

Syntax

```
okv manage-access endpoint-group check-status --endpoint-group endpoint_group_name|--
locator-id UUID
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "check-status",
    "options" : {
      "endpointGroup" : "#VALUE",
      "locatorID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint_group` / `-endpointGroup` or `--locator-id` / `locatorID`	Required	The name of the endpoint group or the locator ID (universally unique ID (UUID)) of the endpoint group that you want to check. The `--locator-id` / `locatorID` is required only if you are using a multi-master cluster environment. You must specify either the `--endpoint-group` / `endpointGroup` value or the `--locator-id` / `locatorID` value, not both. To find existing endpoint groups, run the `okv manage-access endpoint-group list` command. To find the locator ID, check the output from the `okv manage-access endpoint-group create` command that was used to create this endpoint group.

JSON Example

1. Generate JSON input for the `okv manage-access endpoint-group check-status` command:

   ```
   okv manage-access endpoint-group check-status --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `check-status_epg.json`) and then edit it so that you can check the endpoint group's status. Specify either the `endpointGroup` value or the `locatorID` value, but not both:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "endpoint-group",
       "action" : "check-status",
   ```

```
    "options" : {
      "locatorID" : "67E0906F-95EE-4A95-A496-D7DAEA5EDC5F"
    }
  }
}
```

3. Run the `okv manage-access endpoint-group check-status` command using the generated JSON file:

   ```
   okv manage-access endpoint-group check-status --from-json check-status_epg.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "status" : "ACTIVE",
       "endpointGroup" : "EPG_HR"
     }
   }
   ```

   Output includes the name of the endpoint group if the endpoint group object is in `ACTIVE` state. The endpoint group name shown here may be different from what was specified at the endpoint group creation time. If the endpoint groups with the same name are created on multiple cluster nodes, then Oracle Key Vault performs naming conflict resolution and it renames all but one endpoint groups by appending `_OKV`*`node-id`* to the endpoint group name. For example, if you named the endpoint group `EPG_HR`, and there is a naming conflict, then the name could be `EPG_HR_OKV01`.

   On deployments other than multi-master cluster, this command returns `Success` if the endpoint group exists and output does not include entries showing the endpoint group name and its state.

okv manage-access endpoint-group create

The `okv manage-access endpoint-group create` command creates a new endpoint group.

Required Authorization

Key Administrator role or Create Endpoint Group system privilege

Syntax

```
okv manage-access endpoint-group create --endpoint-group endpoint_group_name --
description "endpoint group description" --unique TRUE|FALSE
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "create",
    "options" : {
      "endpointGroup" : "#VALUE",
      "description" : "#VALUE",
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint-group` / `endpointGroup`	Required	Name of the endpoint group. See Naming Guidelines for Objects. To find existing endpoint groups, run the `okv manage-access endpoint-group list` command.
`--description` / `description`	Optional	A user-friendly description of the endpoint group enclosed within double quotation marks
`--unique` / `unique`	Optional	Applies to a multi-master cluster environment only. This `--unique` parameter creates the endpoint group as a unique endpoint group. In a multi-master cluster, it is possible that an endpoint group with the same name could be created from two different nodes. If that happens, then the endpoint group names may conflict. The Oracle Key Vault conflict resolution scheme will keep one endpoint group with the given name and rename other endpoint groups with the conflicting names to a name using this format: *given_epg_name*_OKV*node_id*. Valid settings are as follows: • TRUE appends _OKV*node_id* to the given name and thus prevent the conflict for this wallet name. The endpoint group is immediately usable. • FALSE (default) causes Oracle Key Vault to begin a checking process to find if the endpoint group name is unique. A unique ID is returned. You can use this ID to check the status of the endpoint group creation, whether it is in progress (PENDING) or complete (ACTIVE). If the status is PENDING, then it is not yet usable, so any actions performed on the endpoint group will fail. If the status is ACTIVE, then the endpoint group is usable. To check the status, run the `okv admin endpoint-group check-status` command. If the name that you provided is already used in another node, then the name for this endpoint group will have _OKV*xx* appended to it. For example, if you named the endpoint group epg12, and there is a naming conflict, the name could be EPG12_OKV01.

JSON Example

1. Generate JSON input for the `okv manage-access endpoint-group create` command:

   ```
   okv manage-access endpoint-group create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `create_epg.json`) and then edit it so that you can create the endpoint group:

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "create",
    "options" : {
      "endpointGroup" : "epg_hr",
      "description" : "HR endpoint group",
      "unique" : "FALSE"
    }
  }
}
```

3. Run the `okv manage-access endpoint-group create` command using the generated JSON file:

```
okv manage-access endpoint-group create --from-json create_epg.json
```

Output for a multi-master cluster environment appears similar to the following:

```
{
  "result" : "Success",
  "value" : {
    "status" : "PENDING",
    "locatorID" : "67E0906F-95EE-4A95-A496-D7DAEA5EDC5F"
  }
}
```

You can use the `locatorID` from this output with the `okv manage-access endpoint-group check-status` command to display the current state of the endpoint group object. If the object status is `ACTIVE`, then this command also displays the object name after the conflict-name resolution.

okv manage-access endpoint-group delete

The `okv manage-access endpoint-group delete` command deletes an endpoint group.

Required Authorization

Key Administrator role or the Manage Endpoint Group object privilege for the endpoint group

Syntax

```
okv manage-access endpoint-group delete --endpoint-group endpoint_group_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "delete",
    "options" : {
      "endpointGroup" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint-group` / `endpointGroup`	Required	Name of the endpoint group. To find existing endpoint groups, run the `okv manage-access endpoint-group list` command.

JSON Example

1. Generate JSON input for the `okv manage-access endpoint-group delete` command:

   ```
   okv manage-access endpoint-group delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `delete_epg.json`) and then edit it so that you can delete the endpoint group:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "endpoint-group",
       "action" : "delete",
       "options" : {
         "endpointGroup" : "epg_hr"
       }
     }
   }
   ```

3. Run the `okv manage-access endpoint-group delete` command using the generated JSON file:

   ```
   okv manage-access endpoint-group delete --from-json delete_epg.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv manage-access endpoint-group get

The `okv manage-access endpoint-group get` command retrieves detailed information about an endpoint group, such as its member endpoints and wallet access.

Required Authorization

Key Administrator role or the Manage Endpoint Group object privilege for the endpoint group

Syntax

```
okv manage-access endpoint-group get --endpoint-group endpoint_group_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "get",
    "options" : {
      "endpointGroup" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint-group` / `endpointGroup`	Required	Name of the endpoint group. To find existing endpoint groups, run the `okv manage-access endpoint-group list` command.

JSON Example

1. Generate JSON input for the `okv manage-access endpoint-group get` command:

   ```
   okv manage-access endpoint-group get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_ep_group.json`) and then edit it to specify the endpoint group:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "endpoint-group",
       "action" : "get",
       "options" : {
         "endpointGroup" : "hr_ep_grp"
       }
     }
   }
   ```

3. Run the `okv manage-access endpoint-group get` command using the generated JSON file:

   ```
   okv manage-access endpoint-group get --from-json get_ep_group.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "createdBy" : "OKVADMIN",
       "creationTime" : "2021-07-14 13:09:14",
       "description" : "",
       "endpointGroup" : "HR_EP_GRP",
       "endpointGroupMembers" : [ {
         "description" : "",
   ```

```
      "endpoint" : "HR_DB_EP_1"
    }, {
      "description" : "",
      "endpoint" : "HR_DB_EP_2"
    } ],
    "walletAccess" : [ {
      "access" : "RO_MW",
      "wallet" : "hr_wallet"
    } ]
  }
}
```

okv manage-access endpoint-group list

The `okv manage-access endpoint-group list` command retrieves a list of endpoint groups and their associated information.

Required Authorization

Key Administrator role or the Manage Endpoint Group object privilege for the endpoint group

Syntax

`okv manage-access endpoint-group list --limit` *`number_of_endpoints`*

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "list",
    "options" : {
      "limit" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--limit` / `limit`	Optional	Number of endpoint groups to list. Enter any whole number from 1 and higher. If the limit is specified, then Oracle Key Vault fetches the number of endpoint groups up to the specified limit. If the limit is not specified, then Oracle Key Vault fetches up to 10,000 endpoint groups. If you specify a value that is greater than 10,000, then Oracle Key Vault will attempt to fetch those many endpoint groups, depending on the server, client, and network resources. In the output that you retrieve, the `fetchedObjectCount` value lists the actual number of endpoint groups that are fetched. For example, if you specify `100` as the limit but there are only 50 endpoint groups fetched, then Oracle Key Vault sets `fetchedObjectCount` to `50`. If you omit this parameter, then Oracle Key Vault retrieves up to 10,000 endpoint groups. For another example, if the limit is 100 and `fetchedObjectCount` is `100`, then this means that there are more endpoint groups. To fetch all endpoint groups, you need to run this command with an increased value for the `--limit` parameter. If `fetchedObjectCount` is less than the specified limit, then it means that you have retrieved all the available endpoint groups.

JSON Example

1. Generate JSON input for the `okv manage-access endpoint-group list` command:

   ```
   okv manage-access endpoint-group list --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `list_ep_groups.json`) and then edit it to specify the number of records for the output:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "endpoint-group",
       "action" : "list",
       "options" : {
         "limit" : "3"
       }
     }
   }
   ```

3. Run the `okv manage-access endpoint-group list` command using the generated JSON file:

   ```
   okv manage-access endpoint-group list --from-json list_ep_groups.json
   ```

   Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "endpointGroups" : [ {
      "createdBy" : "OKVADMIN",
      "creationTime" : "2021-07-14 13:09:14",
      "description" : "",
      "endpointGroup" : "EPG_HR"
    }, {
      "createdBy" : "OKVADMIN",
      "creationTime" : "2021-07-16 19:29:03",
      "description" : "",
      "endpointGroup" : "SALES_DB_EPG"
    }, {
      "createdBy" : "OKVADMIN",
      "creationTime" : "2021-07-16 19:29:17",
      "description" : "",
      "endpointGroup" : "ORDERS_DB_EPG"
    } ]
  }
}
```

okv manage-access endpoint-group remove-endpoint

The `okv manage-access endpoint-group remove-endpoint` command removes an endpoint from an endpoint group.

Required Authorization

Key Administrator role or the Manage Endpoint Group object privilege for the endpoint group

Syntax

`okv manage-access endpoint-group remove-endpoint --endpoint-group` *`endpoint_group_name`* `--endpoint` *`endpoint_name`*

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "remove-endpoint",
    "options" : {
      "endpointGroup" : "#VALUE",
      "endpoint" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint-group` / `endpointGroup`	Required	Name of the endpoint group that you want to remove. To find existing endpoints, run the `okv admin endpoint list` command.

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint that is associated with the endpoint group. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv manage-access endpoint-group remove-endpoint` command:

   ```
   okv manage-access endpoint-group remove-endpoint --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `remove_ep_from_epg.json`) and then edit it to remove the endpoint from the endpoint group:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "endpoint-group",
       "action" : "remove-endpoint",
       "options" : {
         "endpointGroup" : "epg_hr",
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv manage-access endpoint-group remove-endpoint` command using the generated JSON file:

   ```
   okv manage-access endpoint-group remove-endpoint --from-json remove_ep_from_epg.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv manage-access endpoint-group update

The `okv manage-access endpoint-group update` command changes the name and description of an endpoint group, and can be used to ensure that the endpoint group name is unique.

Required Authorization

Key Administrator role or the Manage Endpoint Group object privilege for the endpoint group

Syntax

```
okv manage-access endpoint-group update --endpoint-group endpoint_group_name --
description "description" --name new_endpoint_group_name --unique TRUE|FALSE
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "endpoint-group",
    "action" : "update",
    "options" : {
      "endpointGroup" : "#VALUE",
      "name" : "#VALUE",
      "description" : "#VALUE",
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint-group` / `endpointGroup`	Required	Current name of the endpoint group. To find existing endpoint groups, run the `okv manage-access endpoint-group list` command.
`--description` / `description`	Optional	A user-friendly description of the endpoint group enclosed within double quotation marks
`--name` / `name`	Optional	New endpoint group name. See Naming Guidelines for Objects.

Parameter/Template Parameter	Required?	Description
`--unique` / `unique`	Optional	Applies to a multi-master cluster environment only. This `--unique` parameter creates the endpoint group as a unique endpoint group. In a multi-master cluster, it is possible that an endpoint group with the same name could be created from two different nodes. If that happens, then the endpoint group names may conflict. The Oracle Key Vault conflict resolution scheme will keep one endpoint group with the given name and rename other endpoint groups with the conflicting names to a name using this format: *given_epg_name*_OKV*node_id*. Valid settings are as follows: • `TRUE` appends _OKV*node_id* to the given name and thus prevent the conflict for this wallet name. The endpoint group is immediately usable. • `FALSE` (default) causes Oracle Key Vault to begin a checking process to find if the endpoint group name is unique. A unique ID is returned. You can use this ID to check the status of the endpoint group creation, whether it is in progress (`PENDING`) or complete (`ACTIVE`). If the status is `PENDING`, then it is not yet usable, so any actions performed on the endpoint group will fail. If the status is `ACTIVE`, then the endpoint group is usable. To check the status, run the `okv admin endpoint-group check_status` command. If the name that you provided is already used in another node, then the name for this endpoint group will have _OKV*xx* appended to it. For example, if you named the endpoint group `epg12`, and there is a naming conflict, the name could be `EPG12_OKV01`.

JSON Example

1. Generate JSON input for the `okv manage-access endpoint-group update` command:

   ```
   okv manage-access endpoint-group update --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `epg_update.json`) and then edit it so that you can update the endpoint group:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "endpoint-group",
       "action" : "update",
       "options" : {
         "endpointGroup" : "epg_hr",
         "name" : "epg_hr_global",
   ```

```
      "description" : "Global HR Endpoint Group",
      "unique" : "FALSE"
    }
  }
}
```

3. Run the `okv manage-access endpoint-group update` command using the generated JSON file:

   ```
   okv manage-access endpoint-group update --from-json epg_update.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "status" : "PENDING",
       "locatorID" : "67E0906F-95EE-4A95-A496-D7DAEA5EDC5F"
     }
   }
   ```

   This example shows the output for renaming an endpoint group in a multi-master cluster. On renaming, an endpoint group is placed into the `PENDING` state for the duration of the naming conflict resolution.

   You can use the `locatorID` from this output with the `okv manage-access endpoint-group check-status` command to display the current state of the endpoint group object. If the object status is `ACTIVE`, then this command also displays the object name after the conflict-name resolution.

   Unless you renamed the endpoint group in a multi-master cluster, the status and `locatorID` entries are not included in the output.

okv manage-access wallet add-access

The `okv manage-access wallet add-access` command grants an endpoint or an endpoint group a level of access to a wallet.

This command uses a user name and password for the authentication.

Required Authorization

Key Administrator role or manage wallet (`MW`) permission on the wallet

Syntax

```
okv manage-access wallet add-access --wallet wallet_name --endpoint endpoint_name|--
endpoint-group endpoint_group_name --access RO|RM|RO_MW|RM_MW
```

JSON Input File Template

```
{
  "service": {
    "category": "manage-access",
    "resource": "wallet",
    "action": "add-access",
    "options": {
      "wallet": "#VALUE",
      "endpointGroup": "#VALUE",
      "endpoint": "#VALUE",
      "access": "#RO|RM|RO_MW|RM_MW"
    }
```

```
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

`--endpoint` / `endpoint`
or
`--endpoint-group` / `endpointGroup`
	Required	Name of the endpoint or endpoint group. You can only specify either an endpoint or an endpoint group, but not both.
		To find registered endpoints, run the `okv admin endpoint list` command. To find endpoint groups, run `okv manage-access endpoint-group` list.

`--access` / `access`	Required	Enter one of the following values: • `RO` for read-only access • `RM` for read-and-modify access • `RO_MW` for read-only and manage-wallet access • `RM_MW` for read-and-modify and manage-wallet access

JSON Example

1. Generate JSON input for the `okv manage-access wallet add-access` command:

   ```
   okv manage-access wallet add-access --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `add_access_wallet.json`) and then edit it so that you can add wallet access to the endpoint settings or the endpoint group, but not both. The following example shows wallet access to an endpoint only:

   ```
   {
     "service": {
       "category": "manage-access",
       "resource": "wallet",
       "action": "add-access",
       "options": {
         "wallet": "hr_wallet",
         "endpoint": "hr_db_ep",
         "access": "RO"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet add-access` command using the generated JSON file:

   ```
   okv manage-access wallet add-access --from-json add_access_wallet.json
   ```

   Output similar to the following appears:

```
{
  "result": "Success"
}
```

okv manage-access wallet add-object

The `okv manage-access wallet add-object` command adds a security object to a wallet.

This command uses a user name and password for the authentication.

Required Authorization

Key Administrator role or have read-modify permission on the object and manage wallet (`MW`) permission on the wallet.

Syntax

```
okv manage-access wallet add-object --wallet wallet_name --uuid uuid
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "add-object",
    "options" : {
      "wallet" : "#VALUE",
      "uuid" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

JSON Example

1. Generate JSON input for the `okv manage-access wallet add-object` command:

   ```
   okv manage-access wallet add-object --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `add_obj_wallet.json`) and then edit it to specify the object to add to the wallet:

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "add-object",
    "options" : {
      "wallet" : "hr_wallet",
      "uuid" : "7432AED6-6628-4F43-BF7C-9D30023A4301"
    }
  }
}
```

3. Run the `okv manage-access wallet add-object` command using the generated JSON file:

```
okv manage-access wallet add-object --from-json add_object_wallet.json
```

Output similar to the following appears:

```
{
  "result": "Success"
}
```

okv manage-access wallet check-status

The `okv manage-access wallet check-status` command checks the naming conflict resolution status of a wallet in a multi-master cluster.

This command is meant primarily for multi-master cluster environments. However, it is valid for other deployments and can be used to check the existence of a wallet.

This command uses a user name and password for the authentication.

Required Authorization

None, but the user only gets the status for the wallets to which he or she has access.

Syntax

```
okv manage-access wallet check-status --wallet wallet_name|--locator-id UUID
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "check-status",
    "options" : {
      "wallet" : "#VALUE",
      "locatorID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet` or `--locator-id` / `locatorID`	Optional	The name of the wallet or the locator ID (universally unique ID (UUID)) of the wallet that you want to check. The `--locator-id` / `locatorID` is required only if you are using a multi-master cluster environment. You must specify either the `--wallet` / `wallet` value or the `--locator-id` / `locatorID` value, not both. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command. To find the locator ID, check the output of the `okv manage-access wallet create` command that was used to create this wallet.

JSON Example

1. Generate JSON input for the `okv manage-access wallet check-status` command:

   ```
   okv manage-access wallet check-status --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `check_wallet.json`) and then edit it so that you can check the status of the wallet. Specify either the `wallet` value or the `locatorID` value, but not both:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "check-status",
       "options" : {
         "locatorID" : "81800CE6-6AAF-4EF5-A0FD-446ED6625F6A"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet check-status` command using the generated JSON file:

   ```
   okv manage-access wallet check-status --from-json check_wallet.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "status" : "ACTIVE",
       "wallet" : "hr_wallet"
     }
   }
   ```

   Output includes the name of the wallet if the wallet object is in `ACTIVE` state. The wallet name shown here may be different from what was specified at the wallet creation time. If the wallets with the same name are created on multiple cluster nodes, Oracle Key Vault performs naming conflict resolution and it renames all but one wallets by appending

_OKV*node-id* to the wallet name. For example, if you named the wallet HR_WALLET, and there is a naming conflict, the name could be HR_WALLET_OKV01.

On deployments other than multi-master cluster, this command returns Success if the wallet exists and output does not include entries showing the wallet name and its state.

okv manage-access wallet create

The okv manage-access wallet create command creates a wallet.

This command uses a user name and password for the authentication.

Required Authorization

None

Syntax

```
okv manage-access wallet create [--description <description>] [--ssh-server-host-user
<ssh-server-host-user>] [--type <type>] [--unique <unique>] --wallet <wallet>
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "create",
    "options" : {
      "wallet" : "#VALUE",
      "type" : "#GENERAL|SSH_SERVER",
      "description" : "#VALUE",
      "unique" : "#TRUE|FALSE",
      "sshServerHostUser" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
--wallet / wallet	Required	Wallet name. To find the names of existing wallets to which you have access, run the okv manage-access wallet list command. Ensure that you follow the naming guidelines for objects.
--description / description	Optional	A user-friendly description for the wallet, enclosed within double quotation marks

Parameter/Template Parameter	Required?	Description
`--unique` / `unique`	Optional	Applies to a multi-master cluster environment only. This `--unique` parameter creates the wallet as a unique wallet. In a multi-master cluster, it is possible that a wallet with the same name could be created from two different nodes. If that happens, then the wallet names may conflict. The Oracle Key Vault conflict resolution scheme will keep one wallet with the given name and rename other wallets with the conflicting names to a name using this format: *`given_wallet_name`*_OKV*`node_id`*. Valid settings are as follows: • `TRUE` appends _OKV*`node_id`* to the given name and thus prevents the conflict for this wallet name. The wallet is immediately usable. • `FALSE` causes Oracle Key Vault to begin a checking process to find if the wallet name is unique. A unique ID is returned. You can use this ID to check the status of the wallet creation, whether it is in progress (`PENDING`) or complete (`ACTIVE`). If the status is `PENDING`, then it is not yet usable, so any actions performed on the wallet will fail. If the status is `ACTIVE`, then confirm the name of the wallet after Oracle Key Vault performs name resolution for this name by executing the `okv manage-access wallet check-status` command. If the name that you provided is already used in another node, then the name for this wallet will have _OKV*`xx`* appended to it. For example, if you named the wallet `wallet12`, and there is a naming conflict, the name could be `WALLET12_OKV01`. If the name that you provided has no naming conflicts, then it will be accepted as the wallet name without any changes.
`--ssh-server-host-user` / `sshServerHostUser`	Optional	The user on the SSH Server for whom this wallet is intended to authorize SSH access. It can be used only for `SSH_SERVER` type wallet.
`--type` / `<type>`	Optional	The type of wallet to create: • `GENERAL`: Holds any type of security object. This is the default value. • `SSH_SERVER`: Holds SSH public keys and is associated with Oracle Key Vault SSH Server endpoint.

JSON Example

1. Generate JSON input for the `okv manage-access wallet create` command:

   ```
   okv manage-access wallet create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `create_wallet.json`) and then edit it so that you can create the wallet:

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "create",
    "options" : {
      "wallet" : "hr_wallet",
      "description" : "wallet for HR endpoint",
      "unique" : "FALSE"
    }
  }
}
```

3. Run the `okv manage-access wallet create` command using the generated JSON file:

```
okv manage-access wallet create --from-json create_wallet.json
```

Output for a multi-master cluster environment appears as follows:

```
{
  "result" : "Success",
  "value" : {
    "status" : "PENDING",
    "locatorID" : "81800CE6-6AAF-4EF5-A0FD-446ED6625F6A"
  }
}
```

You can use the `locatorID` from this output with the `okv manage-access wallet check-status` command to display the current state of the wallet object. If the object status is `ACTIVE`, then this command also displays the object name after the conflict-name resolution.

Related Topics

- Naming Guidelines for Objects
  The naming guidelines affect the following Oracle Key Vault objects: users, user groups, endpoints, endpoint groups, and virtual wallets.
- okv manage-access wallet check-status
  The `okv manage-access wallet check-status` command checks the naming conflict resolution status of a wallet in a multi-master cluster.

okv manage-access wallet delete

The `okv manage-access wallet delete` command deletes a wallet.

This command uses a user name and password for the authentication.

Required Authorization

Key Administrator role or manage wallet (`MW`) permission on the wallet

Syntax

```
okv manage-access wallet delete --wallet wallet_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "delete",
    "options" : {
      "wallet" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

JSON Example

1. Generate JSON input for the `okv manage-access wallet delete` command:

   ```
   okv manage-access wallet delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `del_wallet.json`) and then edit it to specify the wallet to delete from Oracle Key Vault:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource ": "wallet",
       "action" : "delete",
       "options" : {
         "wallet" : "hr_wallet"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet delete` command using the generated JSON file:

   ```
   okv manage-access wallet delete --from-json del_wallet.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv manage-access wallet get

The `okv manage-access wallet get` command retrieves information about a specified wallet, such as the default wallet name and the wallet access.

This command uses a user name and password for the authentication.

Required Authorization

None

Syntax

```
okv manage-access wallet get --wallet wallet_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "get",
    "options" : {
      "wallet" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

JSON Example

1. Generate JSON input for the `okv manage-access wallet get` command:

   ```
   okv manage-access wallet get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_wallet.json`) and then edit it to specify the name of the wallet:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "get",
       "options" : {
         "wallet" : "hr_wallet"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet get` command using the generated JSON file:

   ```
   okv manage-access wallet get --from-json get_wallet.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
   ```

```
  "value" : {
    "createdBy" : "OKVADMIN",
    "creationTime" : "2025-07-30 19:40:59",
    "description" : "",
    "type" : "GENERAL",
    "wallet" : "hr_wallet",
    "walletAccess" : {
      "endpointAccess" : [ {
        "access" : "RO_MW",
        "defaultWallet" : "",
        "subjectName" : "HR_DB_EP1",
        "type" : "Direct"
      }, {
        "access" : "RO",
        "defaultWallet" : "TRUE",
        "subjectName" : "HR_DB_EP2",
        "type" : "Direct"
      } ],
      "endpointGroupAccess" : [ {
        "access" : "RO_MW",
        "subjectName" : "HR_DB_EPG"
      } ],
      "userAccess" : [ {
        "access" : "RO",
        "subjectName" : "Paul Hill"
      } ],
      "userGroupAccess" : [ {
        "access" : "RO",
        "subjectName" : "HR_GROUP_1"
      } ]
    }
  }
}
```

okv manage-access wallet get-default

The `okv manage-access wallet get-default` command gets the default wallet that has been associated with an endpoint.

This command uses a user name and password for the authentication.

Required Authorization

None, but the default wallet information for the endpoint is returned if the user has some level of access on that wallet.

Syntax

`okv manage-access wallet get-default --endpoint` *`endpoint_name`*

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "get-default",
    "options" : {
      "endpoint" : "#VALUE"
    }
```

```
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv manage-access wallet get-default` command:

   ```
   okv manage-access wallet get-default --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_def_wallet.json`) and then edit it to retrieve the default wallet that is associated with the specified endpoint:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "get-default",
       "options" : {
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet get-default` command using the generated JSON file:

   ```
   okv manage-access wallet get-default --from-json get_def_wallet.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "defaultWallet" : "HR_WALLET"
     }
   }
   ```

okv manage-access wallet list

The `okv manage-access wallet list` command lists wallets on which some level of access is granted to the user.

Required Authorization

None

Syntax

```
okv manage-access wallet list --limit number_of_wallets
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "list",
    "options" : {
      "type" : "#GENERAL|SSH_SERVER",
      "limit" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--limit` / `limit`	Optional	Number of wallets to list. Enter any whole number from 1 and higher. If the limit is specified, then Oracle Key Vault fetches the number of wallets up to the specified limit. If the limit is not specified, then Oracle Key Vault fetches up to 10,000 wallets. If you specify a value that is greater than 10,000, then Oracle Key Vault will attempt to fetch those many wallets, depending on the server, client, and network resources. In the output that you retrieve, the `fetchedObjectCount` value lists the actual number of wallets that are fetched. For example, if you specify `100` as the limit but there are only 50 wallets fetched, then Oracle Key Vault sets `fetchedObjectCount` to `50`. If you omit this parameter, then Oracle Key Vault retrieves up to 10,000 wallets. For another example, if the limit is 100 and `fetchedObjectCount` is `100`, then this means that there are more wallets. To fetch all wallets, you need to run this command with an increased value for the `--limit` parameter. If `fetchedObjectCount` is less than the specified limit, then it means that you have retrieved all the available wallets.
`--type` / `type`	Optional	Type of the wallet. The allowed values are: • `GENERAL` • `SSH_SERVER` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supporting an OR operation.

JSON Example

1. Generate JSON input for the `okv manage-access wallet list` command:

```
okv manage-access wallet list --generate-json-input
```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `list_wallets.json`) and then edit it to specify the number of records:

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "list",
    "options" : {
      "limit" : "3"
    }
  }
}
```

3. Run the `okv manage-access wallet list` command using the generated JSON file:

```
okv manage-access wallet list --from-json list_wallets.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "fetchedObjectCount" : "3",
    "wallets" : [ {
      "createdBy" : "OKVADMIN",
      "creationTime" : "2021-07-13 15:22:02",
      "description" : "",
      "wallet" : "HR_WALLET",
      "type" : "GENERAL"
    }, {
      "createdBy" : "OKVADMIN",
      "creationTime" : "2021-07-30 19:40:59",
      "description" : "",
      "wallet" : "sales_wallet",
      "type" : "GENERAL"
    }, {
      "createdBy" : "OKVADMIN",
      "creationTime" : "2021-09-13 04:55:06",
      "description" : "",
      "wallet" : "ORDERS_WALLET",
      "type" : "GENERAL"
    } ]
  }
}
```

okv manage-access wallet list-endpoint-wallets

The `okv manage-access wallet list-endpoint-wallets` command lists the wallets that are associated with an endpoint.

This command uses a user name and password for the authentication.

Required Authorization

None, but this command returns information about only those wallets on which user has some level of access.

Syntax

```
okv manage-access wallet list-endpoint-wallets --endpoint endpoint_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "list-endpoint-wallets",
    "options" : {
      "endpoint" : "#VALUE",
      "limit" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	The name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate JSON input for the `okv manage-access wallet list-endpoint-wallets` command:

   ```
   okv manage-access wallet list-endpoint-wallets --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `list_ep_wallets.json`) and then edit it so that you can find the wallets that are associated with the specified endpoint:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "list-endpoint-wallets",
       "options" : {
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet list-endpoint-wallets` command using the generated JSON file:

   ```
   okv manage-access wallet list-endpoint-wallets --from-json list_ep_wallets.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
   ```

```
      "wallets" : [ "WALLET10", "WALLET11" ]
    }
  }
```

okv manage-access wallet list-objects

The `okv manage-access wallet list-objects` command retrieves the security objects that are members of the specified wallet.

Required Authorization

The user must have some level of access on the wallet.

Syntax

```
okv manage-access wallet  list-objects [--exclude-wallet-membership  <exclude-wallet-membership>]
[--limit <limit>] [--state  <state>] [--type <type>] --wallet <wallet>
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "list-objects",
    "options" : {
      "wallet" : "#VALUE",
      "state" : "#PRE-ACTIVE|ACTIVE|DEACTIVATED|COMPROMISED|DESTROYED|
DESTROYED_COMPROMISED",
      "type" : "#CERTIFICATE|OPAQUE|PRIVATE_KEY|PUBLIC_KEY|SECRET|SYMMETRIC_KEY",
      "limit" : "#VALUE",
      "excludeWalletMembership" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description
`--limit` / `limit`	Optional	Number of objects to list for the specified wallet. Enter any whole number from 1 and higher. If the limit is specified, then Oracle Key Vault fetches the number of objects up to the specified limit. If the limit is not specified, then Oracle Key Vault fetches up to 10,000 objects. If you specify a value that is greater than 10,000, then Oracle Key Vault will attempt to fetch that value, depending on the server, client, and network resources. In the output that you retrieve, the `fetchedObjectCount` value lists the actual number of objects that are fetched. For example, if you specify `100` as the limit but there are only 50 objects fetched, then Oracle Key Vault sets `fetchedObjectCount` to `50`. If you omit this parameter, then Oracle Key Vault retrieves up to 10,000 objects. For another example, if the limit is 100 and `fetchedObjectCount` is `100`, then this means that there are more objects. To fetch all objects, you need to run this command with an increased value for the `--limit` parameter. If `fetchedObjectCount` is less than the specified limit, then it means that you have retrieved all the available objects.
`--exclude-wallet-membership` / `excludeWalletMembership`	Optional	Controls whether wallet membership information for each object is include in the output. • `TRUE` excludes the wallet membership information for each object. Excluding the wallet membership information may improve the performance of this command if the wallet has large number of objects. • `FALSE` (default) includes the wallet membership information for each object.
`--type` / `type`	Optional	Type of the security object. The allowed values are: • `CERTIFICATE` • `OPAQUE` • `PRIVATE_KEY` • `PUBLIC_KEY` • `SECRET` • `SYMMETRIC_KEY` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supporting an OR operation.

Parameter/Template Parameter	Required?	Description
`--state` / `state`	Optional	State of the security object. The allowed values are: • `PRE-ACTIVE` • `ACTIVE` • `DEACTIVATED` • `COMPROMISED` • `DESTROYED` • `DESTROYED_COMPROMISED` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supporting an OR operation.

JSON Example

1. Generate JSON input for the `okv manage-access wallet list-objects` command:

   ```
   okv manage-access wallet list-objects --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `list_wallet_obj.json`) and then edit it to specify a number of objects for the wallet:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "list-objects",
       "options" : {
         "wallet" : "hr_wallet",
         "limit" : "2",
         "excludeWalletMembership" : "FALSE"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet list-objects` command using the generated JSON file:

   ```
   okv manage-access wallet list-objects --from-json list_wallet_obj.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "fetchedObjectCount" : "2",
       "managedObjects" : [ {
         "creatingEndpoint" : "HR_DB_EP",
         "creationDate" : "2021-07-26 20:19:32",
         "deactivationDate" : "2029-12-25 15:11:11",
         "displayName" : "X.509 Certificate: DN EMAILADDRESS=psmith@example.com,
   CN=vienna, OU=Security, O=Oracle, L=Reston, ST=VA, C=US",
         "name" : "ps1009",
         "protectStopDate" : "2029-12-25 15:11:11",
   ```

```
      "state" : "Pre-Active",
      "type" : "Certificate",
      "uuid" : "975F17DF-11C1-4F16-BFBC-28E9C200C99F",
      "walletMembership" : [ "hr_wallet" ]
    }, {
      "creatingEndpoint" : "EMP_DB_EP",
      "creationDate" : "2021-06-30 21:01:48",
      "deactivationDate" : "",
      "displayName" : "Symmetric Key: Name psc7",
      "name" : "ps100,ps3,psa5,psb6,psc7",
      "protectStopDate" : "",
      "state" : "Active",
      "type" : "Symmetric Key",
      "uuid" : "7432AED6-6628-4F43-BF7C-9D30023A4301",
      "walletMembership" : [ "hr_wallet" ]
    } ]
  }
}
```

okv manage-access wallet list-objects-wallets

Required Authorization

None

Syntax

`okv manage-access wallet list-object-wallets --uuid` *uuid*

JSON Input File Template

```
okv manage-access wallet list-object-wallets --generate-json-input
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "list-object-wallets",
    "options" : {
      "uuid" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object.

JSON Example

1. Generate the JSON input for the `okv manage-access wallet list-objects-wallets` command:

   ```
   okv manage-access wallet list-object-wallets --generate-json-input
   ```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `list_object_by_wallet.json`) and then edit it so that you can find the wallets that are associated with the specified endpoint:

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "list-object-wallets",
    "options" : {
      "uuid" : "23ED3089-B341-43BF-8FB3-432ADFC7511F"
    }
  }
}
```

3. Run the `okv manage-access wallet list-objects-wallets` command using the generated JSON file:

```
okv manage-access wallet list-objects-wallets --from-json list_object_by_wallet.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "wallets" : [ "okv_rac_wallet1" ]
  }
}
```

okv manage-access wallet remove-access

The `okv manage-access wallet remove-access` command removes the access that an endpoint or an endpoint group has to a wallet.

This command uses a user name and password for the authentication.

Required Authorization

Key Administrator role or manage wallet (`MW`) permission on the wallet

Syntax

```
okv manage-access wallet remove-access --wallet wallet_name --endpoint endpoint_name|--
endpoint-group endpoint_group_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "remove-access",
    "options" : {
      "wallet" : "#VALUE",
      "endpointGroup" : "#VALUE",
      "endpoint" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

`--endpoint` / `endpoint`
or
`--endpoint-group` / `endpointGroup`
	Required	Name of the endpoint or endpoint group.
		To find existing endpoints, run the `okv admin endpoint list` command. To find endpoint groups, run `okv manage-access endpoint-group list`.

JSON Example

1. Generate JSON input for the `okv manage-access wallet remove-access` command:

   ```
   okv manage-access wallet remove-access --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**. The JSON input includes wallet access settings for both endpoints and endpoint groups.

2. Save the generated input to a file (for example, `remove_wallet_access_ep.json`) and then edit it to remove wallet access from the endpoint or an endpoint group, but not both. The following example shows how to remove access from an endpoint:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "remove-access",
       "options" : {
         "wallet" : "hr_wallet",
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet remove-access` command using the generated JSON file:

   ```
   okv manage-access wallet remove-access --from-json remove_wallet_access_ep.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv manage-access wallet remove-object

The `okv manage-access wallet remove-object` command removes a security object from a wallet.

This command uses a user name and password for the authentication.

Required Authorization

Key Administrator role or have read-modify permission on the object and manage wallet (`MW`) permission on the wallet.

Syntax

```
okv manage-access wallet remove-object --wallet wallet_name --uuid uuid
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "remove-object",
    "options" : {
      "wallet" : "#VALUE",
      "uuid" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

JSON Example

1. Generate JSON input for the `okv manage-access wallet remove-object` command:

   ```
   okv manage-access wallet remove-object --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `remove_wallet_obj.json`) and then edit it to specify the object to be removed from the wallet:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "remove-object",
       "options" : {
         "wallet" : "hr_wallet",
         "uuid" : "7432AED6-6628-4F43-BF7C-9D30023A4301"
   ```

```
      }
    }
  }
```

3. Run the `okv manage-access wallet remove-object` command using the generated JSON file:

   ```
   okv manage-access wallet remove-object --from-json remove_wallet_obj.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success"
   }
   ```

okv manage-access wallet set-default

The `okv manage-access wallet set-default` command sets the default wallet for an endpoint.

This command uses a user name and password for the authentication.

Required Authorization

Key Administrator role or Manage Endpoint privilege for the endpoint and Full Wallet privileges on the wallet

Syntax

```
okv manage-access wallet set-default --wallet wallet_name --endpoint endpoint_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "set-default",
    "options" : {
      "wallet" : "#VALUE",
      "endpoint" : "#VALUE",
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, run the `okv admin endpoint list` command.

JSON Example

1. Generate the JSON input for the `okv manage-access wallet set-default` command:

   ```
   okv manage-access wallet set-default --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `set_def_wallet.json`) and then edit it to set the default wallet for the endpoint:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "set-default",
       "options" : {
         "wallet" : "hr_wallet",
         "endpoint" : "hr_db_ep"
       }
     }
   }
   ```

3. Run the `okv manage-access wallet set-default` command using the generated JSON file:

   ```
   okv manage-access wallet set-default --from-json set_def_wallet.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv manage-access wallet update

The `okv manage-access wallet update` command updates a wallet.

This command uses a user name and password for the authentication.

Required Authorization

Key Administrator role or manage wallet (`MW`) permission on the wallet

Syntax

```
okv manage-access wallet update --wallet wallet_name --name new_wallet_name --
description description --unique TRUE|FALSE
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "update",
    "options" : {
      "wallet" : "#VALUE",
      "name" : "#VALUE",
      "description" : "#VALUE",
```

```
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.
`--name` / `name`	Optional	A new name for the wallet. See Naming Guidelines for Objects.
`--description` / `description`	Optional	A user-friendly description for the wallet, enclosed within double quotation marks
`--unique` / `unique`	Optional	Applies to a multi-master cluster environment only. This `--unique` parameter creates the wallet as a unique wallet. In a multi-master cluster, it is possible that a wallet with the same name could be created from two different nodes. If that happens, then the wallet names may conflict. The Oracle Key Vault conflict resolution scheme will keep one wallet with the given name and rename other wallets with the conflicting names to a name using this format: *given_wallet_name*_OKV*node_id*. Valid settings are as follows: • TRUE appends _OKV*node_id* to the given name and thus prevent the conflict for this wallet name. The wallet is immediately usable. • FALSE causes Oracle Key Vault to begin a checking process to find if the wallet name is unique. A unique ID is returned. You can use this ID to check the status of the wallet creation, whether it is in progress (PENDING) or complete (ACTIVE). If the status is PENDING, then it is not yet usable, so any actions performed on the wallet will fail. If the status is ACTIVE, then confirm the name of the wallet after Oracle Key Vault performs name resolution for this name by executing the `okv manage-access wallet check-status` command. If the name that you provided is already used in another node, then the name for this wallet will have _OKV*xx* appended to it. For example, if you named the wallet `wallet12`, and there is a naming conflict, the name could be `WALLET12_OKV01`.

JSON Example

1. Generate JSON input for the `okv manage-access wallet update` command:

```
okv manage-access wallet update --generate-json-input
```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `update_wallet.json`) and then edit it to update the name and description of a wallet. This example shows how to update the name of a wallet:

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "update",
    "options" : {
      "wallet" : "hr_wallet",
      "name" : "global_hr_wallet",
      "unique" : "FALSE"
    }
  }
}
```

3. Run the `okv manage-access wallet update` command using the generated JSON file:

```
okv manage-access wallet update --from-json update_wallet.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "status" : "PENDING",
    "locatorID" : "81800CE6-6AAF-4EF5-A0FD-446ED6625F6A"
  }
}
```

This example shows the output for renaming a wallet in a multi-master cluster. On renaming, a wallet is placed into the `PENDING` state for the duration of the naming conflict resolution.

The status and locatorID entries are included in the output only when you rename the wallet in a multi-master cluster.

Related Topics

- okv manage-access wallet check-status
  The `okv manage-access wallet check-status` command checks the naming conflict resolution status of a wallet in a multi-master cluster.

okv manage-access wallet update-access

The `okv manage-access wallet update-access` command updates the level of access that an endpoint or an endpoint group has to a wallet.

This command uses a user name and password for the authentication.

Required Authorization

Key Administrator role or manage wallet (`MW`) permission on the wallet

Syntax

```
okv manage-access wallet update-access --wallet wallet_name --endpoint endpoint_name|--
endpoint-group endpoint_group_name --access RO|RM|RO_MW|RM_MW
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "update-access",
    "options" : {
      "wallet" : "#VALUE",
      "endpointGroup" : "#VALUE",
      "endpoint" : "#VALUE",
      "access" : "#RO|RM|RO_MW|RM_MW"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

`--endpoint` / `endpoint`
or
`--endpoint-group` / `endpointGroup`
	Required	Name of the endpoint or endpoint group. You can only specify either an endpoint or an endpoint group, but not both. To find existing endpoints, run the `okv admin endpoint list` command. To find endpoint groups, run `okv manage-access endpoint-group list`.

`--access` / `access`	Required	Enter one of the following values: • `RO` for read-only access • `RM` for read-and-modify access • `RO_MW` for read-only and manage-wallet access • `RM_MW` for read-and-modify and manage-wallet access

JSON Example

1. Generate JSON input for the `okv manage-access wallet update-access` command:

   ```
   okv manage-access wallet update-access --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**. The JSON input includes wallet access settings for both endpoints and endpoint groups.

2. Save the generated input to a file (for example, `update_wallet_access_ep.json`) and then edit it to update wallet access to an endpoint group or an endpoint, but not both. This example shows how to update access of a wallet to an endpoint:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "update-access",
       "options" : {
         "wallet" : "hr_wallet",
   ```

```
      "endpoint" : "hr_db_ep",
      "access" : "RM"
    }
  }
}
```

3. Run the `okv manage-access wallet update-access` command using the generated JSON file:

   ```
   okv manage-access wallet update-access --from-json update_wallet_access_ep.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

5
Security Object Commands

Endpoints can make use of the security object commands to operate on the managed objects.

- okv managed-object attribute add
  The `okv managed-object attribute add` command adds one or more attributes to a security object.
- okv managed-object attribute delete
  The `okv managed-object attribute delete` command deletes one or more attributes associated with a security object.
- okv managed-object attribute get
  The `okv managed-object attribute get` command retrieves an attribute or list of attributes of a security object.
- okv managed-object attribute get-all
  The `okv managed-object attribute get-all` command retrieves all attributes of a security object.
- okv managed-object attribute list
  The `okv managed-object attribute list` command retrieves the names of attributes associated with a security object.
- okv managed-object attribute modify
  The `okv managed-object attribute modify` command modifies attributes that are associated with a security object.
- okv managed-object certificate get
  The `okv managed-object certificate get` command retrieves a digital certificate.
- okv managed-object certificate register
  The `okv managed-object certificate register` command registers a certificate.
- okv managed-object certificate-request get
  The `okv managed-object certificate-request get` command retrieves a certificate request.
- okv managed-object certificate-request register
  The `okv managed-object certificate-request register` command registers a certificate request object with Oracle Key Vault.
- okv managed-object custom-attribute add
  The `okv managed-object custom-attribute add` command adds a custom attribute to a security object.
- okv managed-object custom-attribute delete
  The `okv managed-object custom-attribute delete` command deletes a custom attribute of a security object.
- okv managed-object custom-attribute modify
  The `okv managed-object custom-attribute modify` command modifies a custom attribute of a security object.
- okv managed-object key create
  The `okv managed-object key create` command creates a symmetric key.

- okv managed-object key get
  The `okv managed-object key get` command retrieves a symmetric key.
- okv managed-object key register
  The `okv managed-object key register` command registers a symmetric key.
- okv managed-object key-pair create
  The `okv managed-object key-pair create` command creates a pair of public and private keys. You can use this command to also create an SSH key pair.
- okv managed-object object activate
  The `okv managed-object object activate` command activates a security object.
- okv managed-object object destroy
  The `okv managed-object object destroy` command requests the server to destroy the key data for a security object.
- okv managed-object object fetch
  The `okv managed-object fetch` command fetches a security object and its attributes together.
- okv managed-object object locate
  The `okv managed-object object locate` command locates a security object.
- okv managed-object object query
  The `okv managed-object object query` command identifies supported operations and objects.
- okv managed-object object revoke
  The `okv managed-object object revoke` command revokes a security object.
- okv managed-object opaque get
  The `okv managed-object opaque get` command retrieves an object that contains opaque data.
- okv managed-object opaque register
  The `okv managed-object opaque register` command registers an opaque security object.
- okv managed-object private-key get
  The `okv managed-object private-key get` command retrieves a private key.
- okv managed-object private-key register
  The `okv managed-object private-key register` command registers a private key. You can use this command to also register an SSH private key.
- okv managed-object public-key get
  The `okv managed-object public-key get` command retrieves a public key.
- okv managed-object public-key register
  The `okv managed-object public-key register` command registers a public key. You can use this command to also register an SSH public key.
- okv managed-object secret get
  The `okv managed-object secret get` command retrieves the secret data from a security object of type secret.
- okv managed-object secret register
  The `okv managed-object secret register` command registers secret data such as passwords or random seeds.
- okv managed-object wallet add-member
  The `okv managed-object wallet add-member` command adds a security object to a wallet as its member.

- okv managed-object wallet delete-member
  The `okv managed-object wallet delete-member` command deletes the membership of the managed-object from a wallet.
- okv managed-object wallet list
  The `okv managed-object wallet list` command lists wallets that have their access granted to the endpoint used to connect to Oracle Key Vault.

okv managed-object attribute add

The `okv managed-object attribute add` command adds one or more attributes to a security object.

To find the existing attributes for the security object, run the `okv managed-object attribute list` command. If you want to create a custom attribute, then use the `okv managed-object custom-attribute add` command.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

```
 okv managed-object attribute add [--activation-date activation date] [--contact-info
contact information] [--deactivation-date deactivation date] [--name name] [--process-
start-date process start date] [--protect-stop-date protect stop date] --uuid uuid
```

You may use the JSON syntax for this command to specify the attributes with the `--uuid` parameter specified at the command line. This is useful for cases where you want to apply the same attribute values to multiple objects. You can re-use the same JSON file and specify different UUIDs at the command line.

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "attribute",
    "action" : "add",
    "options" : {
      "uuid" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss"
      }
    }
  }
}
```

Parameters

Parameter/ Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column. See, okv managed-object object locate

`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format.
		`"activationDate" : "now" --starts immediately`
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		`"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this date and time`
		If you omit this parameter, then the activation date is retrieved from the certificate file that is uploaded.
		`If activationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore < deactivationDate <= notAfter`
		`activationDate < deactivationDat`
		If `activation-date` does not meet the criteria an error message displays.

`contact-info` / `contactInfo`	Optional	The attribute is used for descriptive purposes only.

`--deactivation-date` / `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`. If you omit this parameter, then the deactivation date is retrieved from the certificate file that is uploaded.
		`If deactivationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore <= activationDate < notAfter`
		`activationDate < deactivationDate`
		If `deactivation-date` does not meet the criteria an error message displays.

Parameter/ Template Parameter	Required?	Description

`--name` / `name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

`--process-start-date` / `process start date`	Optional	Specifies the date and time. When a valid security object begin to process cryptographically protected information.
`--protect-stop-date` / `protectStop Date`	Optional	Specifies the date and time, after which a valid security object cannot be used for applying cryptographic protection.

Parameter/ Template Parameter	Required?	Description

/ attributes	Required	A JSON object with the list of attributes. You must use the JSON syntax to add an attribute. You cannot specify attributes at the command line. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command.
		Attributes that you can enter are as follows:
		• `name` includes the following:
		– `value` is the value name.
		– `type` is either `text` or `uri`. The default type is `text` when the `name` attribute is used in the command line.
		• `contactInfo` : Contact information of the object. For example, an e-mail address.
		• The following date and time attributes:
		– `activationDate`: Contains the date and time when using the Managed Object.
		– `deactivationDate`: The date and time when the Managed Object should not be used for any purpose, except for decryption.
		– `protectStopDate`: The date and time when a valid Managed Object is used to process cryptographically protected information, for example, decryption or unwrapping.
		– `processStartDate`: The date and time after which a valid Managed Object should not be used for applying cryptographic protection, for example, encryption or wrapping.
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "now"` *`--starts immediately`*
		`"activationDate" : "now+PT10M"` *`--starts 10 minutes from now`*
		`"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		To display the time in UTC format, use the Linux `date` command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		See Key Management Interoperability Protocol Specification Version 1.1 for more details about these attributes.

JSON Example

1. Generate JSON input for the `okv managed-object attribute add` command:

   ```
   okv managed-object attribute add --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

   .

2. Save the generated input to a file (for example, `add_attribute.json`) and then edit it to include the attributes for the security object. For example:

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "attribute",
    "action" : "add",
    "options" : {
      "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
      "attributes" : {
        "contactInfo" : "psmith@example.com",
        "deactivationDate" : "2024-12-31 09:00:00",
        "name" : {
          "value" : "PROD-HRDB-MKEY",
          "type" : "text"
         },
       "protectStopDate" : "2024-09-30 09:00:00"
      }
    }
  }
}
```

3. Run the `okv managed-object attribute add` command using the generated JSON file:

```
okv managed-object attribute add --from-json add_attribute.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "attributes" : {
      "contactInfo" : "Added",
      "deactivationDate" : "Added",
      "name" : "Added",
      "protectStopDate" : "Added"
    }
  }
}
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object attribute delete

The `okv managed-object attribute delete` command deletes one or more attributes associated with a security object.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

`okv managed-object attribute delete [--contact-info] [--name` *`name`*`] --uuid` *`uuid`*

You may use the JSON syntax for this command to specify the attributes with the `--uuid` parameter specified at the command line. This is useful for cases where you want to apply the

same attribute values to multiple objects. You can re-use the same JSON file and specify different UUIDs at the command line.

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "attribute",
    "action": "delete",
    "options": {
      "uuid": "#VALUE",
      "attributes": {
        "name": {
          "value": "#VALUE"
        },
        "contactInfo": "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--contact-info` / `contactInfo`	Optional	The attribute is used for descriptive purposes only.

`--name` / `name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

Parameter/Template Parameter	Required?	Description
/ attributes	Required	A JSON object with the list of attributes. You must use the JSON syntax to specify the attribute. You cannot specify attributes at the command line. To find the existing attributes for the managed object, run the okv managed-object attribute list command. Attributes that you can delete are as follows: • name (You must also specify the value of the name attribute instance that you want to delete.)

JSON Example

1. Generate JSON input for the okv managed-object attribute delete command:

   ```
   okv managed-object attribute delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, del_attribute.json) and then edit it so that you can delete the attributes associated with a security object:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "attribute",
       "action": "delete",
       "options": {
         "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
         "attributes": {
           "name": {
             "value": "PROD-HRDB-MKEY"
           }
         }
       }
     }
   }
   ```

3. Run the okv managed-object attribute delete command using the generated JSON file:

   ```
   okv managed-object attribute delete --from-json del_attribute.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
       "attributes": {
         "name": "Deleted"
       }
     }
   }
   ```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object attribute get

The `okv managed-object attribute get` command retrieves an attribute or list of attributes of a security object.

Required Authorization

The endpoint must have read permission on the object.

Syntax

```
okv managed-object attribute get [--activation-date] [--archive-date] [--certificate-
length] [--certificate-type]
 [--compromise-date] [--compromise-occurrence-date] [--contact-info] [--crypto-
parameter] [--crypto-usage-mask] [--cryptographic-algorithm]
 [--cryptographic-length][--custom-attribute] [--custom-attributes custom attributes] [--
deactivation-date] [--destroy-date]
 [--digest] [--digital-signing-algorithm ][--extractable] [--initial-date] [--last-
change-date] [--link] [--name][--never-extractable]
 [--object-group-member] [--object-type][--process-start-date] [--protect-stop-date] [--
state] --uuid <uuid> [--x509-certificate-issuer]
 [--x509-certificate-subject]
```

You may use the JSON syntax for this command to specify the attributes with the `--uuid` parameter specified at the command line. This is useful for cases where you want to apply the same attribute values to multiple objects. You can re-use the same JSON file and specify different UUIDs at the command line.

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "attribute",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE",
      "attributes" : [ "#ATTRIBUTE_NAME", "#ATTRIBUTE_NAME", "#ATTRIBUTE_NAME"],
      "customAttributes" : [ "#ATTRIBUTE_NAME", "#ATTRIBUTE_NAME", "#ATTRIBUTE_NAME" ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--activation-date /`	Optional	The date and time when the security object may begin to be used.
`--archive-date /`	Optional	The date and time when the security object was placed in archival storage.
`--certificate-length /`	Optional	The length, in bytes, of the Certificate object.
`--certificate-type /`	Optional	The type of certificate.
`--compromise-date /`	Optional	The date and time when the Managed Cryptographic Object entered the compromised state.

Parameter/Template Parameter	Required?	Description
`--compromise-occurrence-date /`	Optional	The date and time when the security object was first believed to be compromised. Use this setting only when `KEY_COMPROMISE` is specified for the `--code` parameter.
`--contact-info /`	Optional	The attribute is for descriptive purposes only.
`--crypto-parameter /`	Optional	Cryptographic parameters such as `Hashing Algorithm`, `Padding Method`, `Block Cipher Mode`, and `Key Role Type`.
`--crypto-usage-mask /`	Optional	The cryptographic usage of the security object.
`--cryptographic-algorithm/`	Optional	The algorithm used in the security object.
`--cryptographic-length/`	Optional	The length, in bits, of the cryptographic key material of the security object.
`--custom-attributes / customAttributes`	Optional	Additional attributes defined by the endpoint and not interpreted by Oracle Key Vault.

`--deactivation-date /`	Optional	The date and time when the security object must not be used for any purpose. It has the same format as `activation-date`. If you omit this parameter, then the deactivation date is retrieved from the certificate file that is uploaded.
		`If deactivationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore <= activationDate < notAfter`
		`activationDate < deactivationDate`
		If `deactivation-date` does not meet the criteria an error message displays.

`--destroy-date /`	Optional	The date and time when the security object was destroyed.
`--digest /`	Optional	The digest value of the security object.
`--digital-signing-algorithm /`	Optional	The digital signature algorithm associated with a digitally signed object.
`--extractable /`	Optional	One of the following values: • `FALSE`: The server prevent the object value from being retrieved and the server sets the object value. • `TRUE`: If the client does not provides the value or it is not specified in the endpoint configuration. .
`--initial-date/`	Optional	The date and time when the security object was first created or registered at the server.
`--last-change-date /`	Optional	The date and time of the last change of the specified object.
`--link /`	Optional	The link from one security object to another, closely related target security object.
`--name /`	Optional	The name of the object to locate.
`--never-extractable /`	Optional	The value is `TRUE` if the Extractable attribute has always been `FALSE`.

Parameter/Template Parameter	Required?	Description
`--object-group-member /`	Optional	The object group member type: `DEFAULT` or `FRESH`.
`--object-type /`	Optional	The type of security object.
`--process-start-date /`	Optional	The date and time when a valid security object may begin to be used to process cryptographically protected information.
`--protect-stop-date /`	Optional	The date and time after which a valid security object cannot be used for applying cryptographic protection.
`--state /`	Optional	The different states of an object as `PREACTIVE`, `ACTIVE`, `DEACTIVATED`, `COMPROMISED`, `DESTROYED`, and `DESTROYED_COMPROMISED`.
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--attributes`	Optional	Specifies one or more attributes from the following: `activationDate`, `archiveDate`, `certificateLength`, `certificateType`, `compromiseDate`, `compromisOccurrenceDate`, `contactInfo`, `cryptoUsageMaskcustomAttributes`, `deactivationDate`, `destroyDate` `digitalSigningAlgorithm` `initialDate`, `lastChangeDate`, `name`, `neverExtractable`, `objectGroupMember`, `objectType`, `processStartDate`, `protectStopDate`, `state`, `uuid`, `x509CertificateIssuer`, `x509CertificateSubject`.

JSON Example

1. Generate JSON input for the `okv managed-object attribute get` command:

   ```
   okv managed-object attribute get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_attribute.json`) and then edit it to retrieve the attributes associated with the security object:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "attribute",
       "action" : "get",
       "options" : {
         "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
   ```

```
      "attributes": [
        "activationDate",
        "contactInfo",
        "cryptoUsageMask",
        "cryptographicAlgorithm",
        "cryptographicLength",
        "name",
        "objectType",
        "state",
        "extractable",
        "neverExtractable"
      ],
      "customAttributes" : ["x-ApplicationTag"]
    }
  }
}
```

3. Run the `okv managed-object attribute get` command using the generated JSON file:

```
okv managed-object attribute get --from-json get_attribute.json
```

Output similar to the following appears:

```
{
  "result": "Success",
  "value": {
    "attributes": {
      "activationDate": "2020-11-21 01:00:00",
      "contactInfo": "psmith@example.com",
      "cryptoUsageMask": [
        "ENCRYPT",
        "DECRYPT"
      ],
      "cryptographicAlgorithm": "AES",
      "cryptographicLength": "256",
      "extractable" : "false",
      "name": [
        {
          "type": "text",
          "value": "PROD-HRDB-MKEY"
        }
      ],
      "neverExtractable" : "TRUE",
      "objectType": "Symmetric Key",
      "state": "Active"
    },
    "customAttributes": [
      {
        "index": "1",
        "name": "x-ApplicationTag",
        "type": "Text String",
        "value": "HR-Production"
      }
    ]
  }
}
```

The activation date parameter can have one of the following formats:

```
"activationDate" : "now" --starts immediately
"activationDate" : "now+PT10M" --starts 10 minutes from now
"activationDate" : "2021-12-20 10:30:00" --starts at this date and time
```

```
"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this
date and time
```

If you omit this parameter, then the activation date is retrieved from the certificate file that is uploaded.

```
If activationDate meets the date criteria,the date is overwritten with the provided date.
    notBefore < deactivationDate <= notAfter
    activationDate < deactivationDate
```

If `activation-date` does not meet the criteria, an error message displays.

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object attribute get-all

The `okv managed-object attribute get-all` command retrieves all attributes of a security object.

Required Authorization

The endpoint must have read permission on the object.

Syntax

```
okv managed-object attribute get-all --uuid UUID
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "attribute",
    "action" : "get-all",
    "options" : {
      "uuid" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

JSON Example

1. Generate JSON input for the `okv managed-object attribute get-all` command:

```
okv managed-object attribute get-all --generate-json-input
```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_all_attributes.json`) and then edit it to get all the attributes of the security object:

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "attribute",
    "action" : "get-all",
    "options" : {
      "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
    }
  }
}
```

3. Run the `okv managed-object attribute get-all` command using the generated JSON file:

```
okv managed-object attribute get-all --from-json get_all_attributes.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "attributes" : {
      "activationDate" : "2022-10-28 18:58:34",
      "cryptoUsageMask" : [ "SIGN" ],
      "cryptographicAlgorithm" : "RSA",
      "cryptographicLength" : "2048",
      "digest" : {
        "algorithm" : "SHA-256",
        "digestValue" :
"9E6C6718C34FE44E3C91558CB83278CEC0706B8906BF0DED23A6117DC9EE6CD0",
        "keyFormatType" : "RAW"
      },
      "extractable" : "false",
      "fresh" : "Yes",
      "initialDate" : "2022-10-28 18:58:36",
      "lastChangeDate" : "2022-10-28 18:58:38",
      "neverExtractable" : "true",
      "objectType" : "Private Key",
      "processStartDate" : "2022-10-28 18:58:36",
      "state" : "Active",
      "uuid" : "06024832-066A-4F02-BF03-FA5B09E9A6AF"
    }
  }
}
```

okv managed-object attribute list

The `okv managed-object attribute list` command retrieves the names of attributes associated with a security object.

The `okv managed-object attribute list` command shows the key `customAttributes` if the object has one or more custom attributes. To find the custom attributes defined for the object, run the `okv managed-object attribute get-all` command.

Required Authorization

The endpoint must have read permission on the object.

Syntax

```
okv managed-object attribute list --uuid UUID
```

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "attribute",
    "action": "list",
    "options": {
      "uuid": "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

JSON Example

1. Generate JSON input for the `okv managed-object attribute list` command:

   ```
   okv managed-object attribute list --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `list_attributes.json`) and then edit it to retrieve the list of attributes for the security object:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "attribute",
       "action": "list",
       "options": {
         "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
       }
     }
   }
   ```

3. Run the `okv managed-object attribute list` command using the generated JSON file:

   ```
   okv managed-object attribute list --from-json list_attributes.json
   ```

Output similar to the following appears:

```
{
  "result": "Success",
  "value": {
    "attributes" : [
      "activationDate",
      "contactInfo",
      "cryptoUsageMask",
      "cryptographicAlgorithm",
      "cryptographicLength",
      "deactivationDate",
      "digest",
      "extractable",
      "fresh",
      "initialDate",
      "lastChangeDate",
      "name",
      "neverExtractable",
      "objectType",
      "processStartDate",
      "protectStopDate",
      "state",
      "uuid"
      ],
    "customAttributes" : [ "x-ApplicationTag" ]
  }
}
```

okv managed-object attribute modify

The `okv managed-object attribute modify` command modifies attributes that are associated with a security object.

To find the existing attributes for the managed object, run the `okv managed-object attribute list` command.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

```
okv managed-object attribute modify [--activation-date activation date] [--contact-info
contact information]
    [--deactivation-date deactivation date] [--name name] [--process-start-date process
start date]
    [--protect-stop-date protect stop date] --uuid uuid
```

You may use the JSON syntax for this command to specify the attributes with the `--uuid` parameter specified at the command line. This is useful for cases where you want to apply the same attribute values to multiple objects. You can re-use the same JSON file and specify different UUIDs at the command line.

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "attribute",
    "action" : "modify",
```

```
    "options" : {
      "uuid" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "newValue" : "#VALUE",
          "newType" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "extractable" : "#TRUE|FALSE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format.
		`"activationDate" : "now"` *`--starts immediately`*
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		If you omit this parameter, then the activation date is retrieved from the certificate file that is uploaded.
		`If activationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore < deactivationDate <= notAfter`
		`activationDate < deactivationDate`
		If `activation-date` does not meet the criteria an error message displays.

`contact-info` / `contactInfo`	Optional	The attribute is used for descriptive purposes only.

Parameter/Template Parameter	Required?	Description

`--deactivation-date` / `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`. If you omit this parameter, then the deactivation date is retrieved from the certificate file that is uploaded.
		
		`If deactivationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore <= activationDate < notAfter`
		`activationDate < deactivationDate`
		
		If `deactivation-date` does not meet the criteria an error message displays.

`--name` / `name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `newValue`
		• `newType`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "newValue" : "KEY2", "newType" : "text"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

`--process-start-date` / `processStartDate`	Optional	Specifies the date and time. When a valid security object begin to process cryptographically protected information.
`--protect-stop-date` / `protectStopDate`	Optional	Specifies the date and time, after which a valid security object cannot be used for applying cryptographic protection.

Parameter/Template Parameter	Required?	Description
`/attributes`	Required	Attribute names and their values. You must use the JSON syntax to specify the attribute. You cannot specify attributes at the command line. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command. Attributes that you can enter are as follows: • `name` includes the following: – `value` is the existing name value. – `newValue` is the new name value. – `newType` is the new name value type. If you want to change the type only, then you must provide a `value` and `newValue`. The default value is `text`. • `contactInfo` • The following date and time attributes: – `activationDate` – `deactivationDate` – `protectStopDate` – `processStartDate` • `extractable` can be set as follows: – TRUE Allows the object to be extracted from Oracle Key Vault. – FALSE Prevents the key material within the object from being extracted from Oracle Key Vault. However, the metadata of the object (including object attributes, state, and so on) can still be retrieved from Oracle Key Vault. As a user who has the Key Administrator role, you can modify the extractable attribute setting of an existing symmetric key or private key to either TRUE or FALSE. A user with read-modify access on an existing symmetric key can also modify its extractable attribute setting, however, this is allowed only to apply the stricter setting, that is, to set the value to FALSE to make the object non-extractable. Such users cannot modify the extractable attribute setting to make a symmetric key extractable if it is currently non-extractable. You can apply the extractable attribute setting to only symmetric keys, and not to other types of security objects. You can use different ways to set the date and time. Examples are as follows: `"activationDate" : "now"` *`--starts immediately`* `"activationDate" : "now+PT10M"` *`--starts 10 minutes from now`* `"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`* `"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*

Parameter/Template Parameter	Required?	Description

		To display the time in UTC format, use the Linux `date` command. For example:
		
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

JSON Example

1. Generate JSON input for the `okv managed-object attribute modify` command:

   ```
   okv managed-object attribute modify --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `modify_attributes.json`) and then edit it to modify the attributes that are associated with the security object:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "attribute",
       "action" : "modify",
       "options" : {
         "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
         "attributes" : {
           "name" : {
             "value" : "PROD-HRDB-MKEY",
             "newValue" : "PROD-GLOBAL-HRDB-MKEY",
             "newType" : "text"
           },
           "contactInfo" : "jscott@example.com",
           "deactivationDate" : "2024-07-31 09:00:00",
           "protectStopDate" : "2024-04-30 09:00:00",
           "extractable" : "FALSE"
         }
       }
     }
   }
   ```

3. Run the `okv managed-object attribute modify` command using the generated JSON file:

   ```
   okv managed-object attribute modify --from-json modify_attributes.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
       "attributes": {
         "contactInfo": "Modified",
         "deactivationDate": "Modified",
         "name": "Modified",
   ```

```
        "protectStopDate": "Modified",
        "extractable" : "Modified"
      }
    }
  }
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object certificate get

The `okv managed-object certificate get` command retrieves a digital certificate.

Required Authorization

The endpoint must have read permission on the certificate object.

Syntax

`okv managed-object certificate get --output_format <text|json> --uuid UUID [--wrap-key-uuid wrapKeyUUID]`

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "certificate",
    "action": "get",
    "options": {
      "uuid": "#VALUE",
      "wrapKeyUUID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the certificate. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

Parameter/Template Parameter	Required?	Description
`--wrap-key-uuid` / `wrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for wrapping. When specified, the managed object will be retrieved from the KMIP server in a wrapped format using the given key. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are 128, 256.
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

Note

Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object certificate get` command:

   ```
   okv managed-object certificate get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_cert.json`) and then edit it to retrieve the specified certificate:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "certificate",
       "action": "get",
       "options": {
         "uuid": "EEED2C4F-33D7-4F9A-BF02-52DD2225A43A"
       }
     }
   }
   ```

3. Run the `okv managed-object certificate get` command using the generated JSON file:

```
okv managed-object certificate get --from-json get_cert.json
```

Output similar to the following appears:

```
{
  "result": "Success",
  "value": {
    "object": "-----BEGIN CERTIFICATE-----
\nMIIDdzCCAl+gAwIBAgICfVEwDQYJKoZIhvcNAQELBQAwazELMAkGA1UEBhMCdXMx\nEzARB << output
truncated >> AYP\n4vwrDwBdNdGtj36GqjuCpz/xCVM9ieSRxJU8\n-----END CERTIFICATE-----"
  }
}
```

Example Using Text as Output Format

```
okv managed-object certificate get --output_format text --uuid EEED2C4F-33D7-4F9A-
BF02-52DD2225A43A
```

Output

Output similar to the following appears:

```
-----BEGIN CERTIFICATE-----
MIIDfDCCAmSgAwIBAgICVN0wDQYJKoZIhvcNAQELBQAwazELMAkGA1UEBhMCdXMx
EzARBgNVBAgTCkNhbGlmb3JuaWExFTATBgNVBAcMDFJlZHdvb2RfQ2l0eTEPMA0G
A1UEChMGT3JhY2xlMRIwEAYDVQQLDAlLZXlfVmF1bHQxCzAJBgNVBAMTAkNBMB4X
DTI0MDQwMjIxMzg0MloXDTI1MDQwMjIxMzg0MlowczELMAkGA1UEBhMCdXMxEzAR
BgNVBAgTCkNhbGlmb3JuaWExFTATBgNVBAcMDFJlZHdvb2RfQ2l0eTEPMA0GA1UE
ChMGT3JhY2xlMRIwEAYDVQQLDAlLZXlfVmF1bHQxEzARBgNVBAMTCllUS3hVQUNn
UFAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDs/j10tz0hNX7tZQn3
S7I7lA7OkNO0eKiQWUThs/gvE+ARQlHC9xaRIONIFmLN41bQQdOQ4wWLDTv8CIK9
QgxldPdyVoTqV5D6x+kSfqI/BPuWvKyvIXuErr1XM/2LBh63Pu/2DXXDsmZK5BZn
wQrQ4iz/OjA8pG0gpboJJKnvVDzhul0z6lwvuVWLGwTqRwwXovlc/VF5/G5KReW0
stIZAfvpPtG/vlCbMd+LgGGyqUwWmdpTf7s0MMX1+kQ10vT4lV7jBNbLKl8u21bv
0d7Gt5Gqbh1+VLUEU/2tg+G2i8nMI3U0wc1IU2ndNV/YytzILg11AAYWvYikFiui
WwejAgMBAAGjIjAgMB4GA1UdEQQXMBWHBAr0QMaCDTEwLjI0NC42NC4xOTgwDQYJ
KoZIhvcNAQELBQADggEBADRzbI0uxExK2WlNNnFilLH/MmQVa2rMKybCFsiHiNjf
Rseq1Tm+qIYQlosaceIVw/lQLineVt73eXFhS66O01qPk2sG0WGRB08/BiQvDv2V
KwYJ5I9EVzILCmGeMCOLsdbNlodBiDcWBOubbbiyDShStgjEddoiVizfmfjeCUq2
sSN3hr4tgTZupgNA1RB9n+krBrSbR16aMbmrjqEubaBywnQNMiNLid7uRx243p96
WNup2dzGiuTJYyyRtbjpTPz1ZCQhqi5DsVFrclvH2V/EZj4rcLg/BS0cBcEc2gc/
9aYwIPbClN8Cj1bOGBQ66DvXVPBRNMfW5s8cWoX8JU4=
-----END CERTIFICATE-----
```

okv managed-object certificate register

The `okv managed-object certificate register` command registers a certificate.

Required Authorization

None

Syntax

```
okv managed-object certificate register [--activation-date activation date] [--algorithm
algorithm] [--crypto-parameter crypto parameter] [--custom-attribute custom attribute]
[--deactivation-date deactivation date] [--length length] [--mask mask] [--name name] --
```

```
object object [--unwrap-key-uuid unwrapKeyUUID] [--private-key-uuid private-key-uuid] [--
sub-type sub-type] [--type type] [--wallet wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "certificate",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "algorithm" : "#RSA",
      "unwrapKeyUUID" : "#VALUE",
      "length" : "#1024,2048,3072,4096(RSA)",
      "mask" : [ "#VERIFY", "#EXPORT" ],
      "type" : "X_509",
      "subType" : "#USER_CERT|TRUSTPOINT",
      "privateKeyUUID" : "#VALUE",
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ],
      "cryptoParameters" : [ {
        "blockCipherMode" : "#VALUE",
        "paddingMethod" : "#VALUE",
        "hashingAlgorithm" : "#VALUE",
        "keyRoleType" : "#VALUE"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Optional	File path to the certificate object. If no file path is provided, a prompt to supply the object in the command line will be displayed.
`--type` / `type`	Optional	Type of certificate. Enter the following value: `X_509`.
`--sub-type` / `subType`	Optional	Sub-type of the certificate. Choose from the following values: • `USER_CERT` • `TRUSTPOINT`

Parameter/Template Parameter	Required?	Description
`--algorithm` / `algorithm`	Optional	Cryptographic algorithm of the public key contained in the certificate. If you omit this parameter, then the algorithm is retrieved from the certificate file that is being uploaded. Enter the following value: • `RSA`
`--unwrap-key-uuid` / `unwrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for unwrapping. When specified, the provided object is expected to be in a wrapped form. It will be unwrapped in the KMIP server using the specified key and will then be registered. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are `128, 256`.
`--length` / `length`	Optional	Length of the public key contained in the certificate. If you omit this parameter, then the key length is retrieved from the certificate file that is uploaded. Choose from the following values: • `1024` • `2048` • `4096`
`--mask` / `mask`	Optional	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `EXPORT` • `VERIFY` (Default value)
`--private-key-uuid` / `privateKeyUUID`	Optional	Universally unique ID (UUID) of the private key associated with the certificate object. To find the unique identifier for the key, run the `okv manage-access wallet list-objects` command or the `okv admin endpoint list-objects` command.
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Optional	Specifies the name of a security object. The allowed values are : • `value` • `type`: Default value is `text`. `-Support complex data format, name attribute in command line --name '{"value" : "KEY1", "type" : "uri"}'or --name '{"value" : "KEY1", "type" : "text}' -Support simplified data format, name attribute in command line. when type is "text" as a default: --name KEY1`

`--custom-attribute` / `customAttribute`	Optional	Specifies custom defined attribute on security object.
		
		`2017-04-29 18:14:51"}' --custom-attribute '[ { "name": "x-OKV Certificate Expiration Date", "value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"} ] '`
		
		`Support simplified data format, name attribute(single instance), in command line --name KEY1`
		
		`Support simplified data format, custom attribute(multi instance),in commandline --custom-attribute "x-local-name:HR" --custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		If the type is Byte String, the value must be a valid hexadecimal representation.

`--crypto-parameter` / `cryptoParameters`	Optional	Includes cryptographic parameters such as Block Cipher Mode, Padding Method, Hashing Algorithm, and Key Role Type. `--crypto-parameter '{"block-cipher-mode":"CBC","padding-method":"NONE","hashing-algorithm":"MD2","key-role-type":"BDK"}'`

Parameter/Template Parameter	Required?	Description

`--deactivation-date` / `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`. If you omit this parameter, then the deactivation date is retrieved from the certificate file that is uploaded.
		
		`If deactivationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore <= activationDate < notAfter`
		`activationDate < deactivationDate`
		
		If `deactivation-date` does not meet the criteria an error message displays.

`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format.
		
		`"activationDate" : "now"` *`--starts immediately`*
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		
		If you omit this parameter, then the activation date is retrieved from the certificate file that is uploaded.
		
		`If activationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore < deactivationDate <= notAfter`
		`activationDate < deactivationDat`
		
		If `activation-date` does not meet the criteria an error message displays.

Parameter/Template Parameter	Required?	Description

`--attributes/ attributes`	Optional	Attribute names and their values. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command.
		You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax.
		Attributes that you can enter are as follows:
		• `name` includes the following:
		– `value` is the name value.
		– `type` is either `text` or `uri`. The default value is `text`.
		• `contactInfo`
		• The following date and time attributes:
		– `activationDate`
		– `deactivationDate`
		– `protectStopDate`
		– `processStartDate`
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "now" --starts immediately`
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		`"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this date and time`
		To display the time in UTC format, use the Linux `date` command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

CLI Example

```
okv managed-object certificate register --type X_509 --private-key-uuid
95092BD2-B546-4F9A-BF0B-D8ECDC548546 --algorithm RSA --mask "VERIFY" --object
/Users/dopark/test/my.crt --name cert_0701 --activation-date now
--deactivation-date "2030-10-10 10:10:10
```

JSON Example

1. Generate JSON input for the `okv managed-object certificate register` command:

   ```
   okv managed-object certificate register --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `reg_cert.json`) and then edit it to register the specified certificate:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "certificate",
       "action" : "register",
       "options" : {
         "object" : "./cert.pem",
         "algorithm" : "RSA",
         "length" : "2048",
         "mask" : [ "VERIFY"],
         "type" : "X_509",
         "subType" : "USER_CERT",
         "privateKeyUUID" : "D497994E-74CD-4F60-BF7C-52F254142705",
         "wallet" : "hr_wallet",
         "attributes" : {
           "name" : {
             "value" : "FINDB-PROD-CERT",
             "type" : "text"
           },
           "contactInfo" : "psmith@example.com",
           "activationDate" : "2020-12-31 09:00:00",
           "deactivationDate" : "2024-12-31 09:00:00",
           "processStartDate" : "2020-12-31 09:00:00",
           "protectStopDate" : "2024-12-31 09:00:00"
         }
       }
     }
   }
   ```

3. Run the `okv managed-object certificate register` command using the generated JSON file:

   ```
   okv managed-object certificate register --from-json reg_cert.json
   ```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "uuid" : "EEED2C4F-33D7-4F9A-BF02-52DD2225A43A"
  }
}
```

Example Using Text as Output Format

```
okv managed-object certificate register --output_format text --object
certificate_file_path --type certificate_type --sub-type certificate_sub_type --
algorithm cryptographic_algorithm --length key_length --mask cryptographic_usage_mask --
private-key-uuid private_key_uuid  --wallet wallet_name
```

Output

Output similar to the following appears:

```
EEED2C4F-33D7-4F9A-BF02-52DD2225A43A
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object certificate-request get

The `okv managed-object certificate-request get` command retrieves a certificate request.

Required Authorization

The endpoint must have read permission on the certificate request object.

Syntax

```
okv managed-object certificate-request get --output_format <text|json> --uuid UUID [--
wrap-key-uuid wrapKeyUUID]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "certificate-request",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE",
      "wrapKeyUUID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the certificate request. To find the unique identifier for the certificate request, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the **Unique Identifier** column.
`--wrap-key-uuid` / `wrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for wrapping. When specified, the managed object will be retrieved from the KMIP server in a wrapped format using the given key. **Note** Supported wrapping key lengths are AES. Supported wrapping key lengths are `128`, `256`.

Parameter/Template Parameter	Required?	Description
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object certificate-request get` command:

   ```
   okv managed-object certificate-request get  --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_cert_req.json`) and then edit it to specify the UUID of the certificate request:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "certificate-request",
       "action" : "get",
       "options" : {
         "uuid" : "BC0E9004-82E0-4FFA-BFF2-29A67DDD5C64"
       }
     }
   }
   ```

3. Run the `okv managed-object certificate-request get` command using the generated JSON file:

```
okv managed-object certificate-request get  --from-json get_cert_req.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "object" : "-----BEGIN NEW CERTIFICATE REQUEST-----
\nMIIC5TCCAc0CAQAwdDELMAkGA1UEBhMCdXMxEzARBgNVBAgTCkNhbGlmb3JuaWEx << output
truncated >> \nDtWoeZfNYHcWPFmHK8aiLCgzeFG62xRdyg==\n-----END NEW CERTIFICATE
REQUEST-----"
  }
}
```

Example Using Text as Output Format

```
okv managed-object certificate-request get --output_format text --uuid
BC0E9004-82E0-4FFA-BFF2-29A67DDD5C64
```

Output

Output similar to the following appears:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIC6TCCAdECAQAwczELMAkGA1UEBhMCdXMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
FTATBgNVBAcMDFJlZHdvb2RfQ2l0eTEPMA0GA1UEChMGT3JhY2xlMRIwEAYDVQQL
DAlLZXlfVmF1bHQxEzARBgNVBAMTCllUS3hVQUNnUFAwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQDs/j10tz0hNX7tZQn3S7I7lA7OkNO0eKiQWUThs/gv
E+ARQlHC9xaRIONIFmLN41bQQdOQ4wWLDTv8CIK9QgxldPdyVoTqV5D6x+kSfqI/
BPuWvKyvIXuErr1XM/2LBh63Pu/2DXXDsmZK5BZnwQrQ4iz/OjA8pG0gpboJJKnv
VDzhul0z6lwvuVWLGwTqRwwXovlc/VF5/G5KReW0stIZAfvpPtG/vlCbMd+LgGGy
qUwWmdpTf7s0MMX1+kQ10vT4lV7jBNbLKl8u21bv0d7Gt5Gqbh1+VLUEU/2tg+G2
i8nMI3U0wc1IU2ndNV/YytzILg11AAYWvYikFiuiWwejAgMBAAGgMTAvBgkqhkiG
9w0BCQ4xIjAgMB4GA1UdEQQXMBWHBAr0QMaCDTEwLjI0NC42NC4xOTgwDQYJKoZI
hvcNAQELBQADggEBAEkyFWs4O5AL8wbG1fJbfcOiXla9htESxqvxc+13/cpvnh1X
akifbJHIM/KNWtGo7Y4m/9ZZo+gbOQmLQgqaDVBbzlM7e0J0h6q3PITMqS3NyNcd
DS8lHaBX82Gv5MldIZcQ972JGXG/bkMuCy+XxtCTH7n7teFEmMZW3RJ0tGtmE16I
VzrIMcv7R2thKWRKI4YeTEltKukt8nOTfF1xRtfk4i4qyVJDLvOsU7aq+NqnXW3r
i1/SdCTkkP/lej/PovpFlzHQf9dFklc39vqTuHPXpWJGqe1uydhEVODB8DQlS6hf
n94/8z3OB4cLiiZGC4jFoKYBFabGYzhCoDtdckM=
-----END NEW CERTIFICATE REQUEST-----
```

okv managed-object certificate-request register

The `okv managed-object certificate-request register` command registers a certificate request object with Oracle Key Vault.

Required Authorization

None

Syntax

```
okv managed-object certificate-request register [--custom-attribute custom attribute] [--
deactivation-date deactivation date] [--name name] --object object [--unwrap-key-uuid
unwrapKeyUUID] --private-key-uuid private-key-uuid [--type type] [--wallet wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "certificate-request",
    "action" : "register",
    "options" : {
      "type" : "#CRMF,PKCS10,PEM,PGP",
      "object" : "#VALUE",
      "privateKeyUUID" : "#VALUE",
      "wallet" : "#VALUE",
      "unwrapKeyUUID" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Optional	File path to the certificate-request object. If no file path is provided, a prompt to supply the object in the command line will be displayed.
`--unwrap-key-uuid` / `unwrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for unwrapping. When specified, the provided object is expected to be in a wrapped form. It will be unwrapped in the KMIP server using the specified key and will then be registered. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are `128, 256`.

Parameter/Template Parameter	Required?	Description
`--type` / `type`	Optional	Type of certificate request. Choose from the following values: • `CRMF` • `PKCS10` • `PGP` • `PEM` (Default value)
`--private-key-uuid` / `privateKeyUUID`	Required	Universally unique ID (UUID) of the private key associated with the certificate request to be registered. To find the unique identifier for the key, run the `okv manage-access wallet list-objects` command or the `okv admin endpoint list-objects` command.
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description

`/attributes`	Optional	Attribute names and their values. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command.
		You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax.
		Attributes that you can enter are as follows:
		• `name` includes the following:
		– `value` is the name value.
		– `type` is either `text` or `uri`. The default value is `text`.
		• `contactInfo`
		• The following date and time attributes:
		– `activationDate`
		– `deactivationDate`
		– `protectStopDate`
		– `processStartDate`
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "now" --starts immediately`
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		`"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this date and time`
		To display the time in UTC format, use the Linux `date` command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

Parameter/Template Parameter	Required?	Description
--name / name	Optional	Specifies the name of a security object. The allowed values are : • value • type: Default value is text. -Support complex data format, name attribute in command line --name '{"value" : "KEY1", "type" : "uri"}'or --name '{"value" : "KEY1", "type" : "text}' -Support simplified data format, name attribute in command line. when type is "text" as a default: --name KEY1
--custom-attribute/ customAttribute	Optional	Specifies custom defined attribute on security object. 2017-04-29 18:14:51"}' --custom-attribute '[{ "name": "x-OKV Certificate Expiration Date", "value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"}] ' Support simplified data format, name attribute(single instance), in command line --name KEY1 Support simplified data format, custom attribute(multi instance),in commandline --custom-attribute "x-local-name:HR" --custom-attribute ' [“x-local-name:HR”,“x-local-id:100”]' If the type is Byte String, the value must be a valid hexadecimal representation.
--deactivation-date / deactivationDate	Optional	Specifies when to deactivate a security object. It has the same format as activation-date.
--output_format	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is text.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

CLI Example

```
okv managed-object certificate-request register --name "FINDB-PROD-CERTREQ" --custom-
attribute "x-local-name:HR" --activation-date "2020-12-31 09:00:00" --deactivation-date
"2024-12-31 09:00:00
```

JSON Example

1. Generate JSON input for the `okv managed-object certificate-request register` command:

   ```
   okv managed-object certificate-request register --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `reg_cert_req.json`) and then edit it to specify the appropriate certificate request values:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "certificate-request",
       "action" : "register",
       "options" : {
         "type" : "PEM",
         "object" : "./cert_req.pem",
         "privateKeyUUID" : "D497994E-74CD-4F60-BF7C-52F254142705",
         "wallet" : "hr_wallet",
         "attributes" : {
           "name" : {
             "value" : "FINDB-PROD-CERTREQ",
             "type" : "text"
           },
           "contactInfo" : "psmith@example.com",
           "deactivationDate" : "2024-12-31 09:00:00",
           "processStartDate" : "2020-12-31 09:00:00",
           "protectStopDate" : "2024-12-31 09:00:00"
         }
       }
     }
   }
   ```

3. Run the `okv managed-object certificate-request register` command using the generated JSON file:

   ```
   okv managed-object certificate-request register --from-json reg_cert_req.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
   ```

```
        "uuid" : "BC0E9004-82E0-4FFA-BFF2-29A67DDD5C64"
      }
    }
```

Example Using Text as Output Format

```
okv managed-object  certificate-request register --output_format text --object "./
cert_req.pem" --type "PEM" --private-key-uuid "D497994E-74CD-4F60-BF7C-52F254142705" --
wallet hr_wallet  --activation-date now --name hr_csr --custom-attribute "x-local-
name:HR"
```

Output

Output similar to the following appears:

```
BC0E9004-82E0-4FFA-BFF2-29A67DDD5C64
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object custom-attribute add

The `okv managed-object custom-attribute add` command adds a custom attribute to a security object.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

```
okv managed-object custom-attribute add --custom-attribute custom attribute --uuid uuid
```

You may use the JSON syntax for this command to specify the attributes with the `--uuid` parameter specified at the command line. This is useful for cases where you want to apply the same attribute values to multiple objects. You can re-use the same JSON file and specify different UUIDs at the command line.

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "custom-attribute",
    "action" : "add",
    "options" : {
      "uuid" : "#VALUE",
      "customAttribute" : {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--custom-attribute/ customAttribute`	Required	Custom attribute name. Include the prefix `x-` in the attribute name. Do not use the prefix of `x-0KV` with custom attribute names. The custom attributes that start with the `x-0KV` prefix are reserved for use by Oracle Key Vault only. You must use the JSON syntax to specify the attribute. You cannot specify attributes at the command line. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command. You must specify these values for the custom attribute: • `name` is the name of the value that you want to add. • `value` is the value of the attribute. • `type` is `text`, `number`, or `BYTE_STRING`. The value of a Byte String attribute must be a valid hexadecimal representation. See Key Management Interoperability Protocol Specification Version 1.1 for details about JSON attributes.

> **Note**
>
> Use the CLI command syntax to specify the `output_format` option. By default, the output format is JSON. However, the `output_format` option with a value as `text` displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object custom-attribute add` command:

   ```
   okv managed-object custom-attribute add --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `add_cust_attr.json`) and then edit it to include the custom attribute to the security object:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "custom-attribute",
       "action": "add",
   ```

```
    "options": {
      "uuid": "3C695846-BB8D-4FD2-BFC4-E646ACB60404",
      "customAttribute": {
        "name": "x-ApplicationTag",
        "value": "HR-Production",
        "type": "TEXT"
      }
    }
  }
}
```

3. Run the `okv managed-object custom-attribute add` command using the generated JSON file:

   ```
   okv managed-object custom-attribute add --from-json add_cust_attr.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv managed-object custom-attribute delete

The `okv managed-object custom-attribute delete` command deletes a custom attribute of a security object.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

```
 okv managed-object custom-attribute delete --custom-attribute custom attribute --uuid
uuid
```

You may use the JSON syntax for this command to specify the attributes with the `--uuid` parameter specified at the command line. This is useful for cases where you want to apply the same attribute values to multiple objects. You can re-use the same JSON file and specify different UUIDs at the command line.

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "custom-attribute",
    "action" : "delete",
    "options" : {
      "uuid" : "#VALUE",
      "customAttribute" : {
        "name" : "#VALUE",
        "index" : "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--custom-attribute/ customAttribute`	Required	Custom attribute name and its index. Do not use the prefix of `x-OKV` with custom attribute names. The custom attributes that start with the `x-OKV` prefix are reserved for use by Oracle Key Vault only. You must use the JSON syntax to specify the attribute. You cannot specify attributes at the command line. To find the existing attributes for a managed object, run the `okv managed-object attribute get-all` command. You must specify these values for the attribute: • `name` is the name of the value. • `index` is the index of the value. **Note** Do not use the prefix of `x-OKV` with custom attribute names. The custom attributes that start with the `x-OKV` prefix are reserved for use by Oracle Key Vault only. See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

JSON Example

1. Generate JSON input for the `okv managed-object custom-attribute delete` command:

   ```
   okv managed-object custom-attribute delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `del_cust_attr.json`) and then edit it so that you can delete the custom attribute:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "custom-attribute",
       "action" : "delete",
       "options" : {
         "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
   ```

```
        "customAttribute" : {
          "name" : "x-ApplicationTag",
          "index" : "1"
        }
      }
    }
  }
```

3. Run the `okv managed-object custom-attribute delete` command using the generated JSON file:

   ```
   okv managed-object custom-attribute delete --from-json del_cust_attr.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv managed-object custom-attribute modify

The `okv managed-object custom-attribute modify` command modifies a custom attribute of a security object.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

```
 okv managed-object custom-attribute modify --custom-attribute <custom attribute> --uuid
<uuid>
```

You may use the JSON syntax for this command to specify the attributes with the `--uuid` parameter specified at the command line. This is useful for cases where you want to apply the same attribute values to multiple objects. You can re-use the same JSON file and specify different UUIDs at the command line.

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "custom-attribute",
    "action" : "modify",
    "options" : {
      "uuid" : "#VALUE",
      "customAttribute" : {
        "name" : "#VALUE",
        "newValue" : "#VALUE",
        "index" : "#VALUE",
        "type": "#TEXT|NUMBER|BYTE_STRING"
      }
    }
  }
}
```

Parameters

Template Parameter	Required?	Description
`--uuid/ uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`/ customAttribute`	Required	Custom attribute name, value, and index. You must use the JSON syntax to specify the attribute. You cannot specify attributes at the command line. To find the existing attributes for the managed object, run the `okv managed-object attribute get-all` command. You cannot specify attributes at the command line. You must use the JSON syntax to modify a custom attribute. You must specify these values for the attribute: • `name` is the name of the attribute that you want to modify. • `newValue` is the new value for the attribute. • `index` is the index of the attribute that you want to modify. • `type` is `text`, `number`, or `BYTE_STRING`. The value of a Byte String attribute must be a valid hexadecimal representation. Additionally, the `type` is mandatory only if it is Byte String. **Note** Do not use the prefix of `x-OKV` with custom attribute names. The custom attributes that start with the `x-OKV` prefix are reserved for use by Oracle Key Vault only. See Key Management Interoperability Protocol Specification Version 1.1 for details about JSON attributes.

JSON Example

1. Generate JSON input for the `okv managed-object custom-attribute modify` command:

   ```
   okv managed-object custom-attribute modify --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `modify_cust_attr.json`) and then edit it to modify the custom attribute:

   ```
   {
     "service" : {
   ```

```
    "category" : "managed-object",
    "resource" : "custom-attribute",
    "action" : "modify",
    "options" : {
      "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
      "customAttribute" : {
        "name" : "x-ApplicationTag",
        "newValue" : "Global-HR-Production",
        "index" : "1"
      }
    }
  }
}
```

3. Run the `okv managed-object custom-attribute modify` command using the generated JSON file:

   ```
   okv managed-object custom-attribute modify --from-json modify_cust_attr.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv managed-object key create

The `okv managed-object key create` command creates a symmetric key.

Required Authorization

None

Syntax

```
okv managed-object key create [--activation-date activation date] [--algorithm
algorithm] [--crypto-parameter crypto parameter] [--custom-attribute custom attribute][--
deactivation-date deactivation date] [--extractable extractable] [--length length] [--
mask mask] [--name name] [--wallet wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "key",
    "action" : "create",
    "options" : {
      "algorithm" : "#3DES|AES",
      "length" : "#112,168(3DES)|128,192,256(AES)",
      "mask" : [ "#ENCRYPT", "#DECRYPT", "#WRAP_KEY", "#SIGN", "#VERIFY",
                 "#UNWRAP_KEY", "#EXPORT", "#DERIVE_KEY", "#GENERATE_CRYPTOGRAM",
                 "#VALIDATE_CRYPTOGRAM", "#TRANSLATE_ENCRYPT", "#TRANSLATE_DECRYPT",
                 "#TRANSLATE_WRAP", "#TRANSLATE_UNWRAP" ],
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
```

```
        "extractable" : "#TRUE|FALSE"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ],
      "cryptoParameters" : [ {
        "blockCipherMode" : "#VALUE",
        "paddingMethod" : "#VALUE",
        "hashingAlgorithm" : "#VALUE",
        "keyRoleType" : "#VALUE"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--algorithm` / `algorithm`	Optional	Cryptographic algorithm. Choose from the following values: • AES (Default value) • 3DES
`--length` / `length`	Optional	Key length for the algorithm. Choose from the following values: • For AES: 128, 192, 256 (Default value) • For 3DES: 112, 168
`--mask` / `mask`	Optional	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • ENCRYPT (Default value) • DECRYPT (Default value) • SIGN • VERIFY • DERIVE_KEY • EXPORT • GENERATE_CRYPTOGRAM • TRANSLATE_DECRYPT • TRANSLATE_ENCRYPT • TRANSLATE_UNWRAP • TRANSLATE_WRAP • UNWRAP_KEY (Default value) • VALIDATE_CRYPTOGRAM • WRAP_KEY (Default value)
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description
`attributes`	Optional	Sets the `extractable` attribute. • TRUE Allows the object to be extracted from Oracle Key Vault. • FALSE Prevents the key material within the object from being extracted from Oracle Key Vault. However, the metadata of the object (including object attributes, state, and so on) can still be retrieved from Oracle Key Vault. If you do not set the `extractable` attribute, then this value is inherited from the endpoint's configuration. In the command line, you can only specify the `extractable` attribute setting that is stricter than the endpoint's effective setting. For example, you cannot set the `extractable` attribute to TRUE if it is set to FALSE in the endpoint. However, you can always set the extractable attribute value to FALSE in the command line.

`--name/ name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line.`
		`when type is "text" as a default:`
		`--name KEY1`

Parameter/Template Parameter	Required?	Description

`--custom-attribute` / `customAttribute`	Optional	Specifies custom defined attribute on security object.
		
		`2017-04-29 18:14:51"}'`
		`--custom-attribute`
		`'[ { "name": "x-OKV Certificate Expiration Date",`
		`"value" : "2017-04-29 18:14:51", "type" : "text" },`
		`{ "name": "x-local-name", "value" : "HR", "type" : "text"} ] '`
		
		`Support simplified data format, name attribute(single instance), in command line --name KEY1`
		
		`Support simplified data format, custom attribute(multi instance),in commandline`
		`--custom-attribute "x-local-name:HR"`
		`--custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		If the type is Byte String, the value must be a valid hexadecimal representation.

Parameter/Template Parameter	Required?	Description

`--crypto-parameter` / `cryptoParameters`	Optional	Includes cryptographic parameters such as Block Cipher Mode, Padding Method, Hashing Algorithm, and Key Role Type.
		
		`--crypto-parameter '{"block-cipher-mode":"CBC","padding-method":"NONE","hashing-algorithm":"MD2","key-role-type":"BDK"}'`
		
		Allowed values are:
		
		`block-cipher-mode`
		
		• CBC
		• ECB
		• CFB
		• OFB
		• GCM
		
		`padding-method`
		
		• NONE
		• ZEROS
		• PKCS5
		
		`hashing-algorithm`
		
		• MD2
		• MD4
		• MD5
		• SHA_1
		• SHA_224
		• SHA_256
		• SHA_384
		• SHA_512
		• RIPEMD_160
		
		`key-role-type`
		
		• BDK
		• CVK
		• DEK
		• MKAC
		• MKSMC
		• MKSMI
		• MKDAC
		• MKDN
		• MKCP

Parameter/Template Parameter	Required?	Description
		• MKOTH • KEK • MAC16609 • MAC97971 • MAC97972 • MAC97973 • MAC97974 • MAC97975 • ZPK • PVKIBM • PVKPVV • PVKOTH
`--deactivation-date` / `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`.

`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format.
		
		`"activationDate" : "now" --starts immediately`
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		`"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this date and time`

`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

Example without using JSON

```
okv managed-object key create --length 128 --algorithm AES --mask"ENCRYPT" --
name  dw_0706
```

JSON Example

1. Generate JSON input for the `okv managed-object key create` command:

   ```
   okv managed-object key create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `create_key.json`) and then edit it to create the key:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "key",
       "action": "create",
       "options": {
         "algorithm": "AES",
         "length": "256",
         "mask": [
           "ENCRYPT",
           "DECRYPT"
         ],
         "wallet": "hr_wallet",
         "attributes": {
           "extractable" : "FALSE"
         }
       }
     }
   }
   ```

3. Run the `okv managed-object key create` command using the generated JSON file:

   ```
   okv managed-object key create --from-json create_key.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
       "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
     }
   }
   ```

Example Using Text as Output Format

```
okv managed-object key create --output_format text --extractable false --algorithm
cryptographic_algorithm --length key_length --mask crypographic_usage_mask --wallet
wallet_name
```

Output

Output similar to the following appears:

```
2359E04F-DA61-4F7C-BF9F-913D3369A93A
```

okv managed-object key get

The `okv managed-object key get` command retrieves a symmetric key.

Required Authorization

The endpoint must have read permission on the key object.

Syntax

```
okv managed-object key get --output_format text|json --uuid UUID [--wrap-key-uuid wrapKeyUUID]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "key",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE",
      "wrapKeyUUID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the key. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--wrap-key-uuid` / `wrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for wrapping. When specified, the managed object will be retrieved from the KMIP server in a wrapped format using the given key. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are 128, 256.

Parameter/Template Parameter	Required?	Description
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object key get` command:

   ```
   okv managed-object key get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_key.json`) and then edit it to get the specified key:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "key",
       "action" : "get",
       "options" : {
         "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
       }
     }
   }
   ```

3. Run the `okv managed-object key get` command using the generated JSON file:

   ```
   okv managed-object key get --from-json get_key.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
       "object": "E7A641D77DDAF074C62E7A2C2355F2B8D9CD49486E6AF7F38A22CBDEC91630D0"
     }
   }
   ```

   If the symmetric key is not extractable, then the following message appears:

   ```
   {
    "result" : "Failure",
    "message" : "Operation Result Status: Operation Failed, Result Reason:
   ```

```
Not Extractable"
}
```

Example Using Text as Output Format

```
okv managed-object certificate-request get --output_format text --uuid 2359E04F-
DA61-4F7C-BF9F-913D3369A93A
```

Output

Output similar to the following appears:

```
E7A641D77DDAF074C62E7A2C2355F2B8D9CD49486E6AF7F38A22CBDEC91630D0
```

okv managed-object key register

The `okv managed-object key register` command registers a symmetric key.

Required Authorization

None

Syntax

```
okv managed-object key register [--activation-date activation date] [--algorithm
algorithm] [--crypto-parameter crypto parameter] [--custom-attribute custom attribute]
[--deactivation-date deactivation date] [--extractable extractable] [--length length] [--
mask mask] [--name name] --object object [--unwrap-key-uuid unwrapKeyUUID] [--wallet
wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "key",
    "action" : "register",
    "options" : {
      "length" : "#112,168(3DES)|128,192,256(AES)",
      "object" : "#VALUE",
      "algorithm" : "#3DES|AES",
      "unwrapKeyUUID" : "#VALUE",
      "mask" : [ "#ENCRYPT", "#DECRYPT", "#SIGN", "#VERIFY", "#WRAP_KEY", "#UNWRAP_KEY",
           "#EXPORT", "#DERIVE_KEY", "#GENERATE_CRYPTOGRAM", "#VALIDATE_CRYPTOGRAM",
           "#TRANSLATE_ENCRYPT", "#TRANSLATE_DECRYPT", "#TRANSLATE_WRAP",
"#TRANSLATE_UNWRAP" ],
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "extractable" : "#TRUE|FALSE"
      },
      "customAttributes" : [ {
```

```
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ],
      "cryptoParameters" : [ {
        "blockCipherMode" : "#VALUE",
        "paddingMethod" : "#VALUE",
        "hashingAlgorithm" : "#VALUE",
        "keyRoleType" : "#VALUE"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--algorithm` / `algorithm`	Optional	Cryptographic algorithm. Choose from the following values: • AES (Default value) • 3DES
`--unwrap-key-uuid` / `unwrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for unwrapping. When specified, the provided object is expected to be in a wrapped form. It will be unwrapped in the KMIP server using the specified key and will then be registered. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are `128, 256`.
`--length` / `length`	Optional	Key length for the algorithm. Choose from the following values: • For AES: 128, 192, 256 (Default value) • For 3DES: 112, 168

Parameter/Template Parameter	Required?	Description
`--mask` / `mask`	Optional	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` (Default value) • `DECRYPT` (Default value) • `SIGN` • `VERIFY` • `DERIVE_KEY` • `EXPORT` • `GENERATE_CRYPTOGRAM` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `UNWRAP_KEY` (Default value) • `VALIDATE_CRYPTOGRAM` • `WRAP_KEY` (Default value)
`--object` / `object`	Optional	File path to the symmetric key object. If no file path is provided, a prompt to supply the object in the command line will be displayed.
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description

`/attributes`	Optional	Attribute names and their values. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command.
		You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax.
		Attributes that you can enter are as follows:
		• `name` includes the following:
		– `value` is the name value.
		– `type` is either `text` or `uri`.
		• `contactInfo`
		• The following date and time attributes:
		– `activationDate`
		– `deactivationDate`
		– `protectStopDate`
		– `processStartDate`
		• `extractable` can be set as follows:
		– TRUE Allows the object to be extracted from Oracle Key Vault.
		– FALSE Prevents the key material within the object from being extracted from Oracle Key Vault. However, the metadata of the object (including object attributes, state, and so on) can still be retrieved from Oracle Key Vault.
		If you do not set the `extractable` attribute, then this value is inherited from the endpoint's configuration. In the command line, you can only specify the `extractable` attribute setting that is stricter than the endpoint's effective setting. For example, you cannot set the `extractable` attribute to TRUE if it is set to FALSE in the endpoint. However, you can always set the extractable attribute value to FALSE in the command line.
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "now" --starts immediately`
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		`"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this date and time`

Parameter/Template Parameter	Required?	Description

		To display the time in UTC format, use the Linux `date` command. For example:
		
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

`--name/ name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

`--custom-attribute / customAttribute`	Optional	Specifies custom defined attribute on security object.
		
		`2017-04-29 18:14:51"}'`
		`--custom-attribute '[ { "name": "x-OKV Certificate Expiration Date",`
		`"value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"} ] '`
		
		`Support simplified data format, name attribute(single instance), in command line --name KEY1`
		
		`Support simplified data format, custom attribute(multi instance),in commandline`
		`--custom-attribute "x-local-name:HR"`
		`--custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		If the type is Byte String, the value must be a valid hexadecimal representation.

Parameter/Template Parameter	Required?	Description
`--crypto-parameter` / `cryptoParameters`	Optional	Includes cryptographic parameters such as Block Cipher Mode, Padding Method, Hashing Algorithm, and Key Role Type. `--crypto-parameter '{"block-cipher-mode":"CBC","padding-method":"NONE","hashing-algorithm":"MD2","key-role-type":"BDK"}'`
`--deactivation-date`/ `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`.
`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format. `"activationDate" : "now"` *`--starts immediately`* `"activationDate" : "now+PT10M" --starts 10 minutes from now` `"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`* `"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

Example without using JSON

```
      okv managed-object key register
       --length 128 --object/Users/dopark/test/my.key --algorithm AES
       --mask "ENCRYPT" --name dw_0701--activation-date now --deactivation-
date "2030-10-10 10:10:10"
       okv managed-object key register --name
       '{"value" : "dw_key_2"}'--activation-date --deactivation-date
       okv managed-object key register --name
       '{"value" : "dw_key_2", "type" :"uri"}' --activation-date --
deactivation-date
```

```
        okv managed-object key register --name
        '{"value" : "dw_key_2", "type" :"text"}' --activation-date --
deactivation-date
        okv managed-object key register --name
        '{"value" : "dw_key_2", "type" :"uri"}' --custom-attribute
'[ { "name": "x-OKV
        Private Key UID", "value" :"CA8075A4-C13F-4FD0-BF58-FDB984CC879A"},
{ "name":
        "x-NAME2", "value" :"11111"} ] ' --activation-date
        --deactivation-date
```

JSON Example

1. Generate JSON input for the `okv managed-object key register` command:

   ```
   okv managed-object key register --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `reg_key.json`) and then edit it to register the key:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "key",
       "action": "register",
       "options": {
         "length": "256",
         "object": "./object.txt",
         "algorithm": "AES",
         "mask": [
           "ENCRYPT",
           "DECRYPT"
         ],
         "wallet": "hr_wallet",
         "attributes": {
           "name": {
             "value": "FINDB-PROD-MKEY",
             "type": "text"
           },
           "contactInfo" : "pfitch@example.com",
           "activationDate" : "2020-12-31 09:00:00",
           "deactivationDate" : "2024-12-31 09:00:00",
           "processStartDate" : "2020-12-31 09:00:00",
           "protectStopDate" : "2024-12-31 09:00:00",
           "extractable" : "FALSE"
         }
       }
     }
   }
   ```

3. Run the `okv managed-object key register` command using the generated JSON file:

   ```
   okv managed-object key register --from-json reg_key.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
   ```

```
      "uuid": "39BE0215-5D7B-4F38-BF5F-FC87C82AA004"
    }
  }
```

Example Using Text as Output Format

```
okv managed-object key register --output_format text --algorithm cryptographic_algorithm
--length key_length --mask cryptographic_usage_mask --object key_file_path --wallet
wallet_name
```

Output

Output similar to the following appears:

```
39BE0215-5D7B-4F38-BF5F-FC87C82AA004
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object key-pair create

The `okv managed-object key-pair create` command creates a pair of public and private keys. You can use this command to also create an SSH key pair.

Required Authorization

Must be an endpoint

Syntax

```
okv managed-object key-pair create
        --activation-date activation date] [--algorithm algorithm] [--
deactivation-date deactivation date]
       [--length length] [--private-key-custom-attribute private key custom
attribute] [--private-key-extractable private-key-extractable]
       [--private-key-mask private key mask] [--private-key-name private-key-
name] [--public-key-custom-attribute public key custom attribute]
       [--public-key-mask public key mask] [--public-key-name public-key-
name] [--ssh-user ssh-user] [--wallet wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "key-pair",
    "action" : "create",
    "options" : {
      "algorithm" : "#RSA",
      "length" : "#2048,3072,4096(RSA)",
      "sshUser" : "#VALUE",
      "wallet" : "#VALUE",
      "attributes" : {
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss"
      },
      "privateKey" : {
```

```
      "mask" : [
        "#SIGN",
        "#ENCRYPT",
        "#DECRYPT",
        "#WRAP_KEY",
        "#UNWRAP_KEY",
        "#EXPORT",
        "#DERIVE_KEY",
        "#TRANSLATE_ENCRYPT",
        "#TRANSLATE_DECRYPT",
        "#TRANSLATE_WRAP",
        "#TRANSLATE_UNWRAP"
      ],
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "extractable" : "#TRUE|FALSE"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ]
    },
    "publicKey" : {
      "mask" : [
        "#VERIFY",
        "#ENCRYPT",
        "#DECRYPT",
        "#WRAP_KEY",
        "#UNWRAP_KEY",
        "#EXPORT",
        "#DERIVE_KEY",
        "#TRANSLATE_ENCRYPT",
        "#TRANSLATE_DECRYPT",
        "#TRANSLATE_WRAP",
        "#TRANSLATE_UNWRAP"
      ],
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        }
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--algorithm` / `algorithm`	Optional	Cryptographic algorithm. Choose from the following values: • RSA (Default value)
`--length` / `length`	Optional	Key length for the algorithm. Choose from the following values: • For RSA: `2048` (Default value), `3072`, `4096`
`--private-key-mask` / `privateKeyMask`	Optional	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `SIGN` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `UNWRAP_KEY` • `WRAP_KEY` The default values are: • `WRAP_KEY` • `DECRYPT` • `UNWRAP_KEY` • `SIGN`
`--public-key-mask` / `publicKeyMask`	Optional	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DERIVE_KEY` • `EXPORT` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_WRAP` • `TRANSLATE_UNWRAP` • `WRAP_KEY` • `UNWRAP_KEY` • `DECRYPT` • `VERIFY` The default values are: • `ENCRYPT` • `UNWRAP_KEY` • `VERIFY`
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description
`--deactivation-date/ deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as activation-date.

`--activation-date/ activationDate`	Optional	Specifies when to activate a security object. It has the following format.
		
		`"activationDate" : "now" --starts immediately`
		`"activationDate" : "now+PT10M" -- starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		`"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this date and time`

`--private-key-name/ privateKeyName`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "PRIVATE_KEY1", "type" : "uri"}'or`
		`--name '{"value" : " PRIVATE_KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line, when type is "text" as a default:`
		`--name PRIVATE_KEY1`

Parameter/Template Parameter	Required?	Description

`--private-key-custom-attribute/ privateKeyCustomAttributes`	Optional	Specifies custom defined attribute on security object.
		
		`--custom-attribute '[ { "name": "x-OKV Certificate Expiration Date", "value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"} ] '`
		
		`Support simplified data format, name attribute(single instance), in command line --name KEY1`
		
		`Support simplified data format, custom attribute(multi instance),in command line --custom-attribute "x-local-name:HR" --custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		If the type is Byte String, the value must be a valid hexadecimal representation.

`--public-key-name/ publicKeyName`	Optional	Specifies the name of a security object. The allowed values are : • value • type The default value of the type is: • text `-Support complex data format, name attribute in command line --name '{"value" : "PUBLIC_KEY1", "type" : "uri"}'or --name '{"value" : " PUBLIC_KEY1", "type" : "text}' -Support simplified data format, name attribute in command line, when type is "text" as a default: --name PUBLIC_KEY1`

Parameter/Template Parameter	Required?	Description
`--public-key-custom-attribute/ publicKey → customAttributes`	Optional	Specifies when to activate a security object. It has the following format. `Specifies custom defined attribute on security object. 2017-04-29 18:14:51"}' --custom-attribute '[{ "name": "x-OKV Certificate Expiration Date", "value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"}] ' Support simplified data format, name attribute(single instance), in command line --name KEY1 Support simplified data format, custom attribute(multi instance),in command line --custom-attribute "x-local-name:HR" --custom-attribute ' [“x-local-name:HR”,“x-local-id:100”]'` If the type is Byte String, the value must be a valid hexadecimal representation.
`--output_format`	Optional	Specifies the output format of the command. The allowed values are "text" or "json". When the specified output format is "text", the command completes with an exit code 0 upon successful execution, and returns the universally unique IDs (UUIDs) of the private and public keys. The first line of output is the UUID of the private key and the second line is the UUID for the public key. It completes with an exit code of 1 on failure and generates a relevant error message. When the specified output format is "json", the command returns a JSON structure indicating the result of the operation. If the operation is successful, the JSON structure includes the universally unique IDs (UUIDs) of the private and public keys, and completes with an exit code of 0. If the operation fails, the JSON structure includes a relevant error message, and completes with an exit code of 1.
`--ssh-user /sshUser`	Optional	SSH user name. The SSH user is intended to track the actual consumer of the SSH keys, a human, an application, or a machine.

> **Note**
>
> Use the CLI command syntax to specify the `output_format` option. By default, the output format is JSON. However, the `output_format` option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

Example without using JSON

```
okv managed-object key-pair create --length 4096 --algorithm RSA --private-
key-mask "SIGN"
--public-key-mask "VERIFY" --private-key-name 2023_ PRIVATE_KEY_FOR_SIGNING --
public-key-name 2023_PUBLIC_KEY_FOR_VERIFY
```

JSON Example

1. Generate JSON input for the `okv managed-object key-pair create` command:

   ```
   okv managed-object key-pair create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `create_key_pair.json`) and then edit it to create the key pair:

   ```
   {
    "service": {
      "category": "managed-object",
      "resource": "key-pair",
      "action": "create",
      "options": {
        "algorithm": "RSA",
        "length": "3072",
        "privateKey": {
          "mask": ["SIGN", "DECRYPT"],
          "attributes": {
           "extractable" : "FALSE"
          }
        },
        "publicKey": {
          "mask": ["VERIFY", "ENCRYPT"]
        },
        "wallet": "hr_wallet"
      }
    }
   }
   ```

3. Run the `okv managed-object key-pair create` command using the generated JSON file:

   ```
   okv managed-object key-pair create --from-json create_key_pair.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
       "privateKeyUUID": "2BFDEBD7-5AE5-4F31-BFF7-6E8B2D20A170",
       "publicKeyUUID": "6B87CF6A-A10E-4F10-BF06-D92CB0241E8A"
   ```

```
    }
  }
```

Example Using Text as Output Format

```
okv managed-object key-pair create --output_format text --length 4096 --algorithm RSA
 --private-key-mask "SIGN" --public-key-mask "VERIFY"
 --private-key-name 2023_PRIVATE_KEY_FOR_SIGNING –public-key-name
2023_PUBLIC_KEY_FOR_VERIFY
```

Output

Output similar to the following appears:

```
2BFDEBD7-5AE5-4F31-BFF7-6E8B2D20A170
6B87CF6A-A10E-4F10-BF06-D92CB0241E8A
```

okv managed-object object activate

The `okv managed-object object activate` command activates a security object.

See Oasis Key Management Interoperability Protocol Specification Version 1.1 Oasis Standard for various states that a security object can be in.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

```
okv managed-object object activate --uuid UUID
```

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "object",
    "action": "activate",
    "options": {
      "uuid": "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

Parameter/Template Parameter	Required?	Description
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object managed-object activate` command:

   ```
   okv managed-object object activate --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `activate_object.json`) and then edit it to activate the security object:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "object",
       "action": "activate",
       "options": {
         "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
       }
     }
   }
   ```

3. Run the `okv managed-object managed-object activate` command using the generated JSON file:

   ```
   okv managed-object object activate --from-json activate_object.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success"
   }
   ```

Example Using Output Format Text

```
okv managed-object object activate --output_format text --uuid UUID
```

Output

Output similar to the following appears:

- exit code 0 - Indicates Success
- exit code 1- Indicates Failure

okv managed-object object destroy

The `okv managed-object object destroy` command requests the server to destroy the key data for a security object.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

```
okv managed-object object destroy --uuid UUID
```

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "object",
    "action": "destroy",
    "options": {
      "uuid": "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object object destroy` command:

   ```
   okv managed-object object destroy --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `destroy_obj.json`) and then edit it so that you can destroy the security object data:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "object",
       "action": "destroy",
       "options": {
         "uuid": "B36F3AD1-0AC7-4FEB-BF32-79E6F727ECB2"
       }
     }
   }
   ```

3. Run the `okv managed-object object destroy` command using the generated JSON file:

   ```
   okv managed-object object destroy --from-json destroy_obj.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success"
   }
   ```

Example Using Output Format Text

```
okv managed-object object destroy --output_format text --uuid UUID
```

Output

Output similar to the following appears:

- exit code 0 - Indicates Success
- exit code 1- Indicates Failure

okv managed-object object fetch

The `okv managed-object fetch` command fetches a security object and its attributes together.

Required Authorization

The endpoint must have read permission on the object

Syntax

```
 okv managed-object object fetch [--activation-date activation date] [--archive-date
archive date] [--certificate-length certificate length] [--certificate-type certificate
type] [--compromise-date compromise date] [--compromise-occurrence-date compromise
occurrence date] [--contact-info contact information] [--crypto-parameter crypto
parameter][--crypto-usage-mask cryptoUsageMask] [--cryptographic-algorithm cryptographic
algorithm] [--cryptographic-length cryptographic length] [--custom-attribute custom
attribute] [--deactivation-date deactivation date] [--destroy-date destroy date] [--
```

digest *digest*] [--digital-signing-algorithm *digital signing algorithm*] [--extractable *true|false*] [--initial-date *initial date*] [--last-change-date *last change date*] [--link *link*] [--max *max*] [--name *name*] [--never-extractable *true|false*] [--object-group-member *object group member*] [--object-type *object type*] [--process-start-date *process start date*] [--protect-stop-date *protect stop date*] [--single-object *true|false*] [--state *state*] [--x509-certificate-issuer *X.509 certificate issuer*] [--x509-certificate-subject *X.509 certificate subject*]

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "object",
    "action" : "fetch",
    "options" : {
      "max" : "#VALUE",
      "objectGroupMember" : "#FRESH|DEFAULT",
      "attributes" : {
        "name" : {
          "value" : "#VALUE"
        },
        "state" : "#PRE-ACTIVE|ACTIVE|DEACTIVATED|COMPROMISED|DESTROYED|
DESTROYED_COMPROMISED",
        "objectType" : "#VALUE",
        "fresh" : "#YES|NO",
        "objectGroup" : "#VALUE",
        "contactInfo" : "#VALUE",
        "cryptographicAlgorithm" : "#VALUE",
        "cryptographicLength" : "#VALUE",
        "cryptoUsageMask" : "#VALUE",
        "certificateLength" : "#VALUE",
        "certificateType" : "#VALUE",
        "x509CertificateSubject" : "#VALUE",
        "x509CertificateIssuer" : "#VALUE",
        "digitalSigningAlgorithm" : "#VALUE",
        "digest" : {
          "digestValue" : "#VALUE",
          "algorithm" : "#VALUE",
          "keyFormatType" : "#VALUE"
        },
        "link" : {
          "linkType" : "#VALUE",
          "linkValue" : "#VALUE"
        },
        "activationDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "deactivationDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "processStartDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "protectStopDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "initialDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD HH:mm:ss",
        "lastChangeDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "compromiseDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "compromiseOccurrenceDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-
DD HH:mm:ss",
        "destroyDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD HH:mm:ss",
        "archiveDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD HH:mm:ss",
```

```
        "extractable" : "#TRUE|FALSE",
        "neverExtractable" : "#TRUE|FALSE"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ],
      "cryptoParameters" : [ {
        "blockCipherMode" : "#VALUE",
        "paddingMethod" : "#VALUE",
        "hashingAlgorithm" : "#VALUE",
        "keyRoleType" : "#VALUE"
      } ]
    }
  }
}
```

Parameters

Parameter/ Template Parameter	Required?	Description
`--max` / `max`	Optional	Maximum number of objects that this command should return
`--object-group-member` / `objectGroupMember`	Optional	Enter one of the following group values: • `DEFAULT` • `FRESH`
`--state` / `state`	Optional	Enter one of the following states: • `PREACTIVE` • `ACTIVE` • `DEACTIVATED` • `COMPROMISED` • `DESTROYED` • `DESTROYED_COMPROMISED`

Parameter/ Template Parameter	Required?	Description

`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format.
		
		`"activationDate" : "now"` *`--starts immediately`*
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		
		Date attributes support a date range using the format
		`--activation-date "NOW-P90D,NOW"`
		Note: In the range, the start date must be earlier than the end date. The range must follow the ISO 8601 duration format.
		If you omit this parameter, then the activation date is retrieved from the certificate file that is uploaded.
		
		`If activationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore < deactivationDate <= notAfter`
		`activationDate < deactivationDate`
		
		If `activation-date` does not meet the criteria an error message displays.

`--archive-date` / `archiveDate`	Optional	Specifies the date and time of the security object when placed in archival storage.
`--certificate-length` / `certificateLength`	Optional	Specifies the certificate object length in bytes.
`--certificate-type` / `certificateType`	Optional	Specifies the certificate type.
`--compromise-date` / `compromiseDate`	Optional	Specifies the date and time when the Managed Cryptographic Object entered the compromised state.
`--compromise-occurrence-date` / `compromiseOccurrenceDate`	Optional	Specifies the date and time when the security object was first believed to be compromised. Use this setting only when `KEY_COMPROMISE` is specified for the `--code` parameter.
`--contact-info` / `contactInfo`	Optional	The attribute is for descriptive purposes only.

Parameter/ Template Parameter	Required?	Description
`--crypto-usage-mask` / `cryptoUsageMask`	Optional	Specifies the cryptographic usage of the security object.
`--cryptographic-algorithm`/ `cryptographicAlgorithm`	Optional	Specifies the algorithm used in the security object.
`--cryptographic-length` / `cryptographicLength`	Optional	Specifies the length in bits of the cryptographic key material of the security object.

`--deactivation-date` / `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`. If you omit this parameter, then the deactivation date is retrieved from the certificate file that is uploaded.
		`If deactivationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore <= activationDate < notAfter`
		`activationDate < deactivationDate`
		If `deactivation-date` does not meet the criteria an error message displays.

`--destroy-date` / `destroyDate`	Optional	Specifies the date and time when the security object was destroyed.
`--digest` / `digest`	Optional	Specifies the digest value of the security object.
`--digital-signing-algorithm` / `digitalSigningAlgorithm`	Optional	Specifies the digital signature algorithm associated with a digitally signed object.
`--extractable` / `extractable`	Optional	Specifies the true or false values. • FALSE: Specifies the server shall prevent the object value being retrieved. • TRUE: Specifies the default value as true, if the client does not provides the value. .
`--initial-date` / `initialDate`	Optional	Specifies the date and time when the security object was first created or registered at the server.
`--last-change-date` / `lastChangeDate`	Optional	Specifies the date and time of the last change of the specified object.
`--link` / `link`	Optional	Specifies the link from one security object to another, closely related target security object.

Parameter/ Template Parameter	Required?	Description
`--max` / `max`	Optional	Specifies the maximum number of objects that this command should return.
`--object-group-member/ objectGroupMember`	Optional	Specifies the object group member type as `DEFAULT` or `FRESH`.
`--object-type / objectType`	Optional	Specifies the security object type.
`--process-start-date / processStartDate`	Optional	Specifies the date and time when a valid security object start processing the cryptographically protected information.
`--protect-stop-date / protect-stop-date`	Optional	Specifies the date and time after which a valid security object cannot be used for applying cryptographic protection.
`--state / state`	Optional	Specifies the different states of an object as `PREACTIVE`, `ACTIVE`, `DEACTIVATED`, `COMPROMISED`, `DESTROYED`, and `DESTROYED_COMPROMISED`.
`--x509-certificate-issuer / x509CertificateIssuer`	Optional	Specifies the issuer distinguished name in the X.509 certificate.
`--x509-certificate-subject / x509CertificateSubject`	Optional	Specifies the subject distinguished name in the X.509 certificate subject.

Parameter/ Template Parameter	Required?	Description
/attributes	Required	Attributes names and their values of the object to locate. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax. Attributes that you can enter are as follows: • name includes value. • state is the state of the object. • objectType, type of the object. • fresh indicates whether the object is fresh or not. Enter either YES or NO. • objectGroup is the object group or wallet name. • contactInfo is the contact information for the object. • cryptographicAlgorithm is the cryptographic algorithm of the object. • cryptographicLength is the cryptographic length of the object. • cryptoUsageMask is the usage mask of the object. • certificateType is the type of the certificate object. • x509CertificateSubject is the subject of the X.509 certificate. • x509CertificateIssuer is the issuer of the X.509 certificate. • digitalSigningAlgorithm is the digital signature algorithm of the object. • digest is digest of the object, which includes: – digestValue is the value of the digest. – algorithm is the hashing algorithm. – keyFormatType is the format of the object. • link is the link attribute of the object, and it includes: – linkType is the type of the link. – linkValue is the linked object UUID. • The following date and time attributes: – activationDate – deactivationDate – protectStopDate – processStartDate – activationDate – deactivationDate – processStartDate – protectStopDate – initialDate – lastChangeDate – compromiseDate – compromiseOccurrenceDate – destroyDate – archiveDate • extractable can be set as follows: – TRUE Allows the object to be extracted from Oracle Key Vault. – FALSE Prevents the key material within the object from being extracted from Oracle Key Vault. However, the metadata of the object (including object attributes, state, and so on) can still be retrieved from Oracle Key Vault.

Parameter/ Template Parameter	Required?	Description

		• neverExtractable tracks whether the extraction of the security object has always been restricted during its existence in the Oracle Key Vault server. Settings are as follows:
		– TRUE means that the security object was never extractable from Oracle Key Vault during the object's existence.
		– FALSE means the security object was extractable from Oracle Key Vault at least once during the object's existence.
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		To display the time in UTC format, use the Linux date command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`

--custom attributes / customAttribute	Optional	List of custom attributes of the object to locate.
		Custom attributes that you can enter are as follows:
		• name is the name of the custom attribute.
		• value is the value of the custom attribute.
		• type is text, number, or BYTE_STRING.
		`-Specifies custom defined attribute on security object. 2017-04-29 18:14:51"}'`
		`--custom-attribute '[ { "name": "x-OKV Certificate Expiration Date",`
		`"value" : "2017-04-29 18:14:51"},`
		`{ "name": "x-local-name", "value" : "HR"} ] '`
		`-Support simplified data format, name attribute(single instance), in command line --name KEY1`
		`-Support simplified data format, custom attribute(multi instance),in commandline`
		`--custom-attribute "x-local-name:HR"`
		`--custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		The value of a Byte String attribute must be a valid hexadecimal representation.
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

Parameter/ Template Parameter	Required?	Description
`--crypto-parameter` / `cryptoParameters`	Optional	Includes cryptographic parameters such as Block Cipher Mode, Padding Method, Hashing Algorithm, and Key Role Type. `--crypto-parameter '{"block-cipher-mode":"CBC","padding-method":"NONE","hashing-algorithm":"MD2","key-role-type":"BDK"}'`

`--name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`
		The default type:
		• `text`
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as `text` displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the command:

   ```
   okv managed-object object fetch --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `fetch_obj.json`.
3. Run the command using the generated JSON file. For example:

   ```
   okv managed-object object fetch --from-json fetch_obj.json
   --object-group-member --single-object --state
   ```

   Output similar to the following appears:

   ```
   {
   ```

```
  "result" : "Success",
  "value" : [ {
    "attributes" : {
      "activationDate" : "2022-07-01 15:54:38",
      "cryptographicAlgorithm" : "RSA",
      "cryptographicLength" : "2048",
      "cryptoUsageMask" : [ "ENCRYPT" ],
      "deactivationDate" : "2030-10-10 10:10:10",
      "digest" : {
        "digestValue" :
"B8ACE70487179C70DF3A6D320CA0D52FF7F4FB2D9E41E9542E7D8C0166B3D93",
        "keyFormatType" : "RAW",
        "algorithm" : "SHA-256"
      },
      "fresh" : "No",
      "initialDate" : "2022-07-01 15:54:38",
      "lastChangeDate" : "2022-07-01 17:57:24",
       "name" : [ {
        "type" : "text",
        "value" : "private_0701"
      } ],
      "objectType" : "Private Key",
      "processStartDate" : "2022-07-01 15:54:38",
      "state" : "Active",
      "uuid" : "95092BD2-B546-4F9A-BF0B-D8ECDC548546"
    }, "customAttribute" : [ {
      "name" : "x-NAME",
      "index" : "0",
      "type" : "Text String",
      "value" : "test4"
    }, {
      "name" : "x-ID",
      "index" : "0",
      "type" : "Integer",
      "value" : "1"
    } ],
       "object" : "-----BEGIN OPENSSH PRIVATE KEY-----
\nb3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn\nNhAAAAAwEAA
QAAAYEAyYrcnHs6I51lHheg90qTripWIuVKszoluqnBG0+QRLdLKOMIJjajygXKlT\n04DJSrQliR45oki2s/
dgyfsTqpvanjTj7W10O5X2poohlGojumNOmb2p52em55yABUcYOb\nK4Qf5sX4vDpc/
iUQAAABFkb3BhcmtAZG9wYXJrLW1hYw==\n-----END OPENSSH PRIVATE KEY-----\n"
  } ]
}
```


Example Using Output Format Text

```
okv managed-object object fetch --max max_value --object-group-member
object_group_member_type --state state_value --name name_value
```

okv managed-object object locate

The `okv managed-object object locate` command locates a security object.

Required Authorization

The endpoint must have read permission on the objects.

Syntax

```
 okv managed-object object locate [--activation-date activation date] [--archive-date
archive date] [--certificate-length certificate length] [--certificate-type certificate
```

type] [--compromise-date *compromise date*] [--compromise-occurrence-date *compromise occurrence date*] [--contact-info *contact information*] [--crypto-parameter *crypto parameters*] [--crypto-usage-mask *cryptoUsageMask*] [--cryptographic-algorithm *cryptographic algorithm*] [--cryptographic-length *cryptographic length*] [--custom-attribute *custom attribute*] [--deactivation-date *deactivation date*] [--destroy-date *destroy date*] [--digest *digest*] [--digital-signing-algorithm *digital signing algorithm*] [--extractable *true*|*false*] [--initial-date *initial date*] [--last-change-date *last change date*] [--link *link*] [--max *max*] [--name *name*] [--never-extractable *true*|*false*] [--object-group-member *object group member*] [--object-type *object type*] [--process-start-date *process start date*] [--protect-stop-date *protect stop date*] [--single-object *true*|*false*] [--state *state*] [--x509-certificate-issuer *X.509 certificate issuer*] [--x509-certificate-subject *X.509 certificate subject*]

JSON Input File Template

```
okv managed-object object locate --generate-json-input

{
  "service" : {
    "category" : "managed-object",
    "resource" : "object",
    "action" : "locate",
    "options" : {
      "max" : "#VALUE",
      "objectGroupMember" : "#FRESH|DEFAULT",
      "attributes" : {
        "name" : {
          "value" : "#VALUE"
        },
        "state" : "#PRE-ACTIVE|ACTIVE|DEACTIVATED|COMPROMISED|DESTROYED|
DESTROYED_COMPROMISED",
        "objectType" : "#VALUE",
        "fresh" : "#YES|NO",
        "objectGroup" : "#VALUE",
        "contactInfo" : "#VALUE",
        "cryptographicAlgorithm" : "#VALUE",
        "cryptographicLength" : "#VALUE",
        "cryptoUsageMask" : "#VALUE",
        "certificateLength" : "#VALUE",
        "certificateType" : "#VALUE",
        "x509CertificateSubject" : "#VALUE",
        "x509CertificateIssuer" : "#VALUE",
        "digitalSigningAlgorithm" : "#VALUE",
        "digest" : {
          "digestValue" : "#VALUE",
          "algorithm" : "#VALUE",
          "keyFormatType" : "#VALUE"
        },
        "link" : {
          "linkType" : "#VALUE",
          "linkValue" : "#VALUE"
        },
        "activationDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "deactivationDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "processStartDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "protectStopDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "initialDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD HH:mm:ss",
        "lastChangeDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
```

```
HH:mm:ss",
        "compromiseDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD
HH:mm:ss",
        "compromiseOccurrenceDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-
DD HH:mm:ss",
        "destroyDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD HH:mm:ss",
        "archiveDate" : "#YYYY-MM-DD HH:mm:ss|YYYY-MM-DD HH:mm:ss,YYYY-MM-DD HH:mm:ss",
        "extractable" : "#TRUE|FALSE",
        "neverExtractable" : "#TRUE|FALSE"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ],
      "cryptoParameters" : [ {
        "blockCipherMode" : "#VALUE",
        "paddingMethod" : "#VALUE",
        "hashingAlgorithm" : "#VALUE",
        "keyRoleType" : "#VALUE"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--max` / `max`	Optional	Maximum number of objects that this command should return
`--object-group-member` / `objectGroupMember`	Optional	Enter one of the following group values: • `DEFAULT` • `FRESH`
`--state` / `state`	Optional	Enter one of the following states: • `PREACTIVE` • `ACTIVE` • `DEACTIVATED` • `COMPROMISED` • `DESTROYED` • `DESTROYED_COMPROMISED`
`--name` / `name`	Optional	Name of the object to locate.

Parameter/Template Parameter	Required?	Description
`attributes`	Required	Attributes names and their values of the object to locate. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax. Attributes that you can enter are as follows: • `name` includes `value`. • `state` is the state of the object. • `objectType`, type of the object. • `fresh` indicates whether the object is fresh or not. Enter either `YES` or `NO`. • `objectGroup` is the object group or wallet name. • `contactInfo` is the contact information for the object. • `cryptographicAlgorithm` is the cryptographic algorithm of the object. • `cryptographicLength` is the cryptographic length of the object. • `cryptoUsageMask` is the usage mask of the object. • `certificateType` is the type of the certificate object. • `x509CertificateSubject` is the subject of the X.509 certificate. • `x509CertificateIssuer` is the issuer of the X.509 certificate. • `digitalSigningAlgorithm` is the digital signature algorithm of the object. • `digest` is digest of the object, which includes: – `digestValue` is the value of the digest. – `algorithm` is the hashing algorithm. – `keyFormatType` is the format of the object. • `link` is the link attribute of the object, and it includes: – `linkType` is the type of the link. – `linkValue` is the linked object UUID. • The following date and time attributes: – `activationDate` – `deactivationDate` – `protectStopDate` – `processStartDate` – `activationDate` – `deactivationDate` – `processStartDate` – `protectStopDate` – `initialDate` – `lastChangeDate` – `compromiseDate` – `compromiseOccurrenceDate` – `destroyDate`

Parameter/Template Parameter	Required?	Description

		– archiveDate
		• extractable can be set as follows:
		– TRUE Allows the object to be extracted from Oracle Key Vault.
		– FALSE Prevents the key material within the object from being extracted from Oracle Key Vault. However, the metadata of the object (including object attributes, state, and so on) can still be retrieved from Oracle Key Vault.
		• neverExtractable tracks whether the extraction of the security object has always been restricted during its existence in the Oracle Key Vault server. Settings are as follows:
		– TRUE means that the security object was never extractable from Oracle Key Vault during the object's existence.
		– FALSE means the security object was extractable from Oracle Key Vault at least once during the object's existence.
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		To display the time in UTC format, use the Linux date command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`

Parameter/Template Parameter	Required?	Description

`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format.
		
		`"activationDate" : "now"` *`--starts immediately`*
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		
		Date attributes support a date range using the format `--activation-date "NOW-P90D,NOW"`
		Note: In the range, the start date must be earlier than the end date. The range must follow the ISO 8601 duration format.
		If you omit this parameter, then the activation date is retrieved from the certificate file that is uploaded.
		
		`If activationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore < deactivationDate <= notAfter`
		`activationDate < deactivationDat`
		
		If `activation-date` does not meet the criteria an error message displays.

`--archive-date` / `archiveDate`	Optional	Specifies the date and time of the security object when placed in archival storage.
`--certificate-length` / `certificateLength`	Optional	Specifies the certificate object length in bytes.
`--certificate-type` / `certificateType`	Optional	Specifies the certificate type.
`--compromise-date` / `compromiseDate`	Optional	Specifies the date and time when the Managed Cryptographic Object entered the compromised state.
`--compromise-occurrence-date` / `compromiseOccurrence-date`	Optional	Specifies the date and time when the security object was first believed to be compromised. Use this setting only when `KEY_COMPROMISE` is specified for the `--code` parameter.
`--contact-info` / `contactInformation`	Optional	The attribute is for descriptive purposes only.
`--crypto-usage-mask` / `cryptoUsageMask`	Optional	Specifies the cryptographic usage of the security object.
`--cryptographic-algorithm`/ `cryptographicAlgorithm`	Optional	Specifies the algorithm used in the security object.
`--cryptographic-length` / `cryptographicLength`	Optional	Specifies the length in bits of the cryptographic key material of the security object.

Parameter/Template Parameter	Required?	Description
`--custom-attribute` / `customAttribute`	Optional	Specifies the endpoint defined additional attributes that Oracle Key Vault cannot interpret.
`--crypto-parameter` / `cryptoParameters`	Optional	Includes cryptographic parameters such as Block Cipher Mode, Padding Method, Hashing Algorithm, and Key Role Type. `--crypto-parameter '{"block-cipher-mode":"CBC","padding-method":"NONE","hashing-algorithm":"MD2","key-role-type":"BDK"}'`

`--deactivation-date` / `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`. If you omit this parameter, then the deactivation date is retrieved from the certificate file that is uploaded.
		
		`If deactivationDate meets the date criteria,the date is overwritten with the provided date.`
		`notBefore <= activationDate < notAfter`
		`activationDate < deactivationDate`
		
		If `deactivation-date` does not meet the criteria an error message displays.

`--destroy-date`/ `destroyDate`	Optional	Specifies the date and time when the security object was destroyed.
`--digest` / `digest`	Optional	Specifies the digest value of the security object.
`--digital-signing-algorithm` / `digital signing algorithm`	Optional	Specifies the digital signature algorithm associated with a digitally signed object.
`--extractable` / `extractable`	Optional	Specifies the true or false values. • FALSE: Specifies the server shall prevent the object value being retrieved. • TRUE: Specifies the default value as true, if the client does not provides the value. .
`--initial-date` / `initialDate`	Optional	Specifies the date and time when the security object was first created or registered at the server.
`--last-change-date`/ `lastChangeDate`	Optional	Specifies the date and time of the last change of the specified object.
`--link`/ `link`	Optional	Specifies the link from one security object to another, closely related target security object.
`--max`/ `max`	Optional	Specifies the maximum number of objects that this command should return.
`--name`/ `name`	Optional	Specifies the name of the object to locate.
`--never-extractable`/ `neverExtractable`	Optional	Specifies the value as `TRUE` if the Extractable attribute is always `FALSE`.
`--object-group-member`/ `objectGroupMember`	Optional	Specifies the object group member type as `DEFAULT` or `FRESH`.

Parameter/Template Parameter	Required?	Description
`--object-type / objectType`	Optional	Specifies the security object type.
`--process-start-date/ processStartDate`	Optional	Specifies the date and time when a valid security object start processing the cryptographically protected information.
`--protect-stop-date/ protectStopDate`	Optional	Specifies the date and time after which a valid security object cannot be used for applying cryptographic protection.
`--state/ state`	Optional	Specifies the different states of an object as `PREACTIVE`, `ACTIVE`, `DEACTIVATED`, `COMPROMISED`, `DESTROYED`, and `DESTROYED_COMPROMISED`.
`--x509-certificate-issuer/ x509CertificateIssuer`	Optional	Specifies the issuer distinguished name in the X.509 certificate.
`--x509-certificate-subject / x509CertificateSubject`	Optional	Specifies the subject distinguished name in the X.509 certificate subject.

`--custom-attribute / customAttribute`	Optional	List of custom attributes of the object to locate.
		Custom attributes that you can enter are as follows:
		• `name` is the name of the custom attribute.
		• `value` is the value of the custom attribute.
		• `type` is `text`, `number`, or `BYTE_STRING`.
		
		`-Support simplified data format, name attribute(single instance), in command line --name KEY1`
		`-Support simplified data format, custom attribute(multi instance),in commandline`
		`--custom-attribute "x-local-name:HR"`
		`--custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		The value of a Byte String attribute must be a valid hexadecimal representation.
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

Parameter/Template Parameter	Required?	Description

`--name/ name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object object locate` command:

   ```
   okv managed-object object locate --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `locate_obj.json`) and then edit it to locate the security object:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "object",
       "action" : "locate",
       "options" : {
         "max" : "10",
         "objectGroupMember" : "FRESH",
         "attributes" : {
   ```

```
            "state": "ACTIVE",
             "name": {
               "value": "key8"
            },
            "fresh" : "Yes",
            "activationDate": "2021-04-10 07:16:00",
             "link" : {
               "linkType" : "Replaced Object Link",
               "linkValue" : "6B13B7B3-BE61-4FF6-BFB0-4108231392F8"
             },
             "extractable" : "FALSE",
             "neverExtractable" : "TRUE"
         },
        "customAttributes" : [{
           "name": "x-test_1",
           "value": "test_1",
           "type": "TEXT"
          },
          {
           "name": "x-number",
           "value": "1",
           "type": "NUMBER"
          }]
      }
    }
}
```

3. Run the `okv managed-object object locate` command using the generated JSON file:

   ```
   okv managed-object object locate --from-json locate_obj.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "uuids" : [ "6C51CC04-BFA5-4FBD-BFB4-12DCCECAA355" ]
     }
   }
   ```

Example Using Output Format Text

```
okv managed-object object locate --output_format text --max max_value --object-group-
member object_group_member_type --state state_value --name name_value --custom-attribute
custom_attributes_value
```

Output

Output similar to the following appears:

```
6C51CC04-BFA5-4FBD-BFB4-12DCCECAA355
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object object query

The `okv managed-object object query` command identifies supported operations and objects.

Required Authorization

None

Syntax

```
okv managed-object object query
```

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "object",
    "action": "query"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv managed-object object query` command:

   ```
   okv managed-object object query --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `query_obj.json`).

3. Run the `okv managed-object object query` command using the generated JSON file:

   ```
   okv managed-object object query --from-json query_obj.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "objects" : [ "Symmetric Key", "Template", "Secret Data", "Opaque Object",
   "Certificate", "Private Key", "Public Key" ],
       "operations" : [ "Create", "Register", "Re-key", "Locate", "Check", "Get", "Get
   Attributes", "Get Attribute List", "Add Attribute", "Modify Attribute", "Delete
   Attribute", "Activate", "Revoke", "Destroy", "Query", "Discover Versions",
   "Encrypt", "Decrypt", "Sign", "Signature Verify", "Create Key Pair" ]
     }
   }
   ```

okv managed-object object revoke

The `okv managed-object object revoke` command revokes a security object.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

```
okv managed-object object revoke --code code --reason reason --compromise-occurrence-
date date --uuid UUID
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "object",
    "action" : "revoke",
    "options" : {
      "code" : "#UNSPECIFIED|KEY_COMPROMISE|CA_COMPROMISE|AFFILIATION_CHANGED|SUPERSEDED|
CESSATION_OF_OPERATION|PRIVILEGE_WITHDRAWN",
      "reason" : "#VALUE",
      "compromiseOccurrenceDate" : "#YYYY-MM-DD HH:mm:ss",
      "uuid" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--code` / `code`	Optional	Enter one of the following values: • `AFFILIATION_CHANGED` • `CA_COMPROMISE` • `CESSATION_OF_OPERATION` • `PRIVILEGE_WITHDRAWN` • `SUPERSEDED` • `UNSPECIFIED` • `KEY_COMPROMISE` (Default value)
`--reason` / `reason`	Required	Description of the reason for the revocation
`--compromise-occurrence-date` / `compromiseOccurrenceDate`	Optional	Date the compromise took place. This setting is used only if `KEY_COMPROMISE` is selected for the `--code` / `code` parameter.
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

Parameter/Template Parameter	Required?	Description
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object object revoke` command:

   ```
   okv managed-object object revoke --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `revoke_obj.json`) and then edit it so that you can revoke the security object privileges:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "object",
       "action": "revoke",
       "options": {
         "code": "KEY_COMPROMISE",
         "reason": "security incidence",
         "compromiseOccurrenceDate": "2020-11-20 10:34:29",
         "uuid": "E4CA6A16-B3CD-4F98-BF25-4A0EF482B8B8"
       }
     }
   }
   ```

3. Run the `okv managed-object object revoke` command using the generated JSON file:

   ```
   okv managed-object object revoke --from-json revoke_obj.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success"
   }
   ```

Example Using Output Format Text

```
okv managed-object object revoke --output_format text --code code --reason reason --
compromise-occurrence-date date --uuid UUID
```

Output

Output similar to the following appears:

- exit code 0 - Indicates Success
- exit code 1- Indicates Failure

okv managed-object opaque get

The `okv managed-object opaque get` command retrieves an object that contains opaque data.

Required Authorization

The endpoint must have read permission on the object.

Syntax

```
okv managed-object opaque get ----output_format <text|json> --uuid UUID [--wrap-key-uuid wrapKeyUUID]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "opaque",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE",
      "wrapKeyUUID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

Parameter/Template Parameter	Required?	Description
`--wrap-key-uuid` / `wrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for wrapping. When specified, the managed object will be retrieved from the KMIP server in a wrapped format using the given key. **Note** Supported algorithm is AES, Supported key lengths are `128, 256`.
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is: • `text`

Note

Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object opaque get` command:

   ```
   okv managed-object opaque get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_opaque_object.json`) and then edit it to retrieve data from the opaque object:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "opaque",
       "action" : "get",
       "options" : {
         "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
       }
     }
   }
   ```

3. Run the `okv managed-object opaque get` command using the generated JSON file:

   ```
   okv managed-object opaque get --from-json get_opaque_object.json
   ```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "object" :
"2D2D2D2D2D424547494E2050524956415445204B45592D2D2D2D2D0A4D4949457651494241444414E4267
6B71686B6947397730424151454641415343424B637767675363 <<<< Output Truncated>>>>
706753317063363465 6D3630686C72336B786C593858665734317A594A450A724546334C652F4A4F4B496
8674A754C367352734C67553D0A2D2D2D2D2D454E44205052495641544 5204B45592D2D2D2D2D0A"
  }
}
```

Example Using Output Format Text

```
okv managed-object opaque get --output_format text --uuid 2359E04F-DA61-4F7C-
BF9F-913D3369A93A
```

Output

Output similar to the following appears:

```
2D2D2D2D2D424547494E2050524956415445204B45592D2D2D2D2D0A4D4949457651494241444414E42676B716
86B6947397730424151454641415343424B637767675363
<<<< Output Truncated>>>>
706753317063363465 6D3630686C72336B786C593858665734317A594A450A724546334C652F4A4F4B4968674
A754C367352734C67553D0A2D2D2D2D2D454E44205052495641544 5204B45592D2D2D2D2D0A
```

okv managed-object opaque register

The `okv managed-object opaque register` command registers an opaque security object.

Objects containing opaque data are not necessarily interpreted by the server.

Required Authorization

None

Syntax

```
okv managed-object opaque register [--custom-attribute custom attribute] [--name name] --
object object [--unwrap-key-uuid unwrapKeyUUID] [--wallet wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "opaque",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "unwrapKeyUUID" : "#VALUE",
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss"
```

```
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Optional	File path to the opaque object. If no file path is provided, a prompt to supply the object in the command line will be displayed.
`--unwrap-key-uuid` / `unwrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for unwrapping. When specified, the provided object is expected to be in a wrapped form. It will be unwrapped in the KMIP server using the specified key and will then be registered. **Note** Supported wrapping key algorithm is AES, Supported wrapping key lengths are 128, 256.
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description

/attributes	Optional	Attribute names and their values. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command.
		You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax.
		Attributes that you can enter are as follows:
		• `name` includes the following:
		– `value` is the name value.
		– `type` is either `text` or `uri`. The default value is `text`.
		• `contactInfo`
		• The following date and time attributes:
		– `activationDate`
		– `deactivationDate`
		– `protectStopDate`
		– `processStartDate`
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "now"` *`--starts immediately`*
		`"activationDate" : "now+PT10M"` *`--starts 10 minutes from now`*
		`"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		To display the time in UTC format, use the Linux `date` command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

Parameter/Template Parameter	Required?	Description

`--name` / `name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `KEY1`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

`--custom-attribute` / `customAttribute`	Optional	Specifies custom defined attribute on security object.
		
		`Support simplified data format, custom attribute(multi instance),in commandline`
		`--custom-attribute "x-local-name:HR"`
		`--custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		`Support simplified data format, name attribute(single instance), in command line --name KEY1`
		
		Specifies custom defined attribute on security object.
		
		`2017-04-29 18:14:51"}'`
		`--custom-attribute '[ { "name": "x-OKV Certificate Expiration Date",`
		`"value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"} ] '`
		
		If the type is Byte String, the value must be a valid hexadecimal representation.

Parameter/Template Parameter	Required?	Description
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the `output_format` option. By default, the output format is JSON. However, the `output_format` option with a value as `text` displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object opaque register` command:

   ```
   okv managed-object opaque register --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `reg_opaque.json`) and then edit it to register the opaque key:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "opaque",
       "action": "register",
       "options": {
         "object": "./key.pem",
         "wallet": "hr_wallet",
         "attributes": {
           "name": {
             "value": "Opaque-Key-102",
             "type": "text"
           },
           "contactInfo" : "psmith@example.com",
           "processStartDate" : "2020-12-31 09:00:00",
           "protectStopDate" : "2024-12-31 09:00:00"
         }
       }
     }
   }
   ```

3. Run the `okv managed-object opaque register` command using the generated JSON file:

   ```
   okv managed-object opaque register --from-json reg_opaque.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
   ```

```
    "value" : {
      "uuid" : "B44A99FD-F892-4F3E-BF7D-487B68159CC3"
    }
  }
```

Example Using Output Format Text

```
okv managed-object opaque register --output_format text --object object_name --wallet
wallet_name
```

Output

Output similar to the following appears:

```
B44A99FD-F892-4F3E-BF7D-487B68159CC3
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.
- okv managed-object opaque get
  The `okv managed-object opaque get` command retrieves an object that contains opaque data.

okv managed-object private-key get

The `okv managed-object private-key get` command retrieves a private key.

Required Authorization

The endpoint must have read permission on the private key.

Syntax

```
okv managed-object private-key get  --output_format text|json --uuid UUID [--wrap-key-
uuid wrapKeyUUID]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "private-key",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE",
      "wrapKeyUUID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the private key. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--wrap-key-uuid` / `wrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for wrapping. When specified, the managed object will be retrieved from the KMIP server in a wrapped format using the given key. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are 128, 256.

Parameter/Template Parameter	Required?	Description
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. **Note** Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command. The default value is: • `text`

JSON Example

1. Generate JSON input for the `okv managed-object private-key get` command:

   ```
   okv managed-object private-key get  --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_private_key.json`) and then edit it to specify the UUID of the private key:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "private-key",
       "action" : "get",
       "options" : {
         "uuid" : "2F9E2A31-D15A-4F5B-BFA0-761892021DBE"
       }
     }
   }
   ```

3. Run the `okv managed-object private-key get` command using the generated JSON file:

```
okv managed-object private-key get --from-json get_private_key.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "object" : "-----BEGIN PRIVATE KEY-----
\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAg << output truncated >> /onTXJKf8A1kZwPW/
Qa6IpPOGCfOJDtyM9F5X9REaJQr+1\nXw1sBm1Tjh4z/m6rsKK6A4YP\n-----END PRIVATE KEY-----"
  }
}
```

Example Using Output Format Text

`okv managed-object private-key get --output_format text --uuid` *`2F9E2A31-D15A-4F5B-BFA0-761892021DBE`*

Output

Output similar to the following appears:

```
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQC0ZPZQ9CDr4zB6
KVawjB22jmAj8cDnuxfOBCIPSbPrrLTdIPP5nIrbzRjkmXW6tYIWE0Mo4J40NZBH
sOKvgffEFDPYeyJ8MIDfscgRg+7TCVlX3ggZi/FqlOtZLKh9sOhRByHZoaQGt7KB
vwPwOXKb6go4FSSt+Q9+/kfphlc0/pTeR1MH3lZKyQHd3toLG8RrRe4uiWutKsvl
+5bb6+ar4Unlm+mhlbJo0jsW2+l2ecTGFbk0DCXQmc+IXsedFj4Q25poSN/QP9Mm
VFQx7pkLT1zi9ldeGNaenkqBimW5QW4aiERMQy5YYLEf4YukUxAeJkSKG6oTkIcj
3CyWPeLTAgMBAAECggEAGtCDI1IWmVm3pAEEZ3vRvYgjrFQhi5CY9OunsG674DAC
Nnh71kJkYFB3/yBR4ANeUdpiekS2iiYyyyVKR4e3L53yfZ8CvO5syajgh2dSwcCQ
V5T5I+r/Dgu9gWTiVW9NfnXj8eNVbBY0N5W63SN88MYfkhSP043DtZR7VgVnwx2b
Dcq66mkcC1Gv+R3J48hi1vYjfdR97K6mYIsUylqyP0WWP2r8Xaq0EBlRr783L2Ek
n0y22dIg37nlXmE0SHqjYUmBQw19UOI8JPseNaWC+kEzX4KDlTOspfPhcbtlPc8K
9MA9n9kBLGqSTXU1q5xPVmmCvQ3u/VQkhLkk/Y6FwQKBgQDDG47X8byKPPDSEIc2
9kzS53vXV0lL07wd2hTpUPRbmJheBUShEJ1wZXVPO8sF2XCJh5q0DWqJ4EaqgKzI
xeUyqqLknJ5y8veXQsjNZlcdpHcJF6khqn4ImCSxK4fL1N1eHx/es7dsBQssHS8p
U8AAAZuMolsDxpuVU9SblxqMoQKBgQDssd0w+fFhI+8+Pj4cgDzndVhplcXR+wBr
weEE6JywrmVXawjhuoqGPz4TV6XdpmZ1xeMAbJYQLsuza1RoYoYzVh2X61sWojOY
dHLWQFIvjuykVDetGku31Fap9pHLn7m4X6yN7ED+wYB1N1ZjGVzyhSCDQzqpR99t
G5uRWJqm8wKBgEI70LouLjsfhQFK6xwVcO8H9lqcHx3Gu03HAuJRwSJ3IwKH90xW
P5wwVSqYucs6jaGijzogc9JqeJLZK87cn3obw0CoioGrSbHn0hMoctp1QpOrnJb5
Md/pBku+41Km6J9TIZaQm+vnqmtHXQNCkyNtaEjNtOgLPDXtR3eqTDEBAoGAcXSY
Z/+Gul91MEq5jwPRLclLxeCq4AoGm6BlcermILKXlR2TNlAqUktCQBdFREec2Vsv
gmaKT9t+fNQ3s4l2dmkS6l1AXksFnFBFM3nFa1KBEAvmXC08mjNID1RHcmlvpncF
yUmzpemu34DVAc+IhB9YBFNXbQksiaowGa11BA0CgYAI7jtelkB8pucPoYWAT360
HBk+05MXWJ8zei5MAXQgnDLOUV74TpH8GejzyVpoXVnYa65ItSaEO97mUrMuC7Px
qIxgruRJXPtBbffWdt53Px2ZMBCWkwcdWW8Ptd4oK0qa0bM5QeLNOn6vMAEnAbN1
NEsyZz5Ey2oECx1hmz/d9Q==
-----END PRIVATE KEY-----
```

okv managed-object private-key register

The `okv managed-object private-key register` command registers a private key. You can use this command to also register an SSH private key.

Required Authorization

None

Syntax

```
okv managed-object private-key register [--activation-date activation date] [--algorithm
algorithm] [--crypto-parameter crypto parameter] [--custom-attribute custom attribute]
[--deactivation-date deactivation date] [--extractable extractable] --length length [--
mask mask] [--name name] --object object [--unwrap-key-uuid unwrapKeyUUID] [--ssh-user
ssh-user] [--wallet wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "private-key",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "algorithm" : "#RSA",
      "unwrapKeyUUID" : "#VALUE",
      "length" : "#1024,2048,4096(RSA)",
      "sshUser" : "#VALUE",
      "mask" : [ "#SIGN", "#ENCRYPT", "#DECRYPT", "#WRAP_KEY", "#UNWRAP_KEY",
                 "#EXPORT", "#DERIVE_KEY", "#TRANSLATE_ENCRYPT", "#TRANSLATE_DECRYPT",
                 "#TRANSLATE_WRAP", "#TRANSLATE_UNWRAP" ],
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "extractable" : "#TRUE|FALSE"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ],
      "cryptoParameters" : [ {
        "blockCipherMode" : "#VALUE",
        "paddingMethod" : "#VALUE",
        "hashingAlgorithm" : "#VALUE",
        "keyRoleType" : "#VALUE"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Optional	File path to the private-key object. If no file path is provided, a prompt to supply the object in the command line will be displayed.

Parameter/Template Parameter	Required?	Description
`--algorithm` / `algorithm`	Optional	Cryptographic algorithm. The default value is RSA.
`--unwrap-key-uuid` / `unwrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for unwrapping. When specified, the provided object is expected to be in a wrapped form. It will be unwrapped in the KMIP server using the specified key and will then be registered. **Note** SSupported wrapping key algorithm is AES. Supported wrapping key lengths are `128, 256`.
`--length` / `length`	Required	Key length for the algorithm. Choose from the following values: • `1024` • `2048` • `4096`
`--mask` / `mask`	Optional	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `SIGN` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `WRAP_KEY` • `UNWRAP_KEY` The default values are: • `DECRYPT` • `UNWRAP_KEY` • `WRAP_KEY` • `SIGN`
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description

/attributes	Optional	Attribute names and their values. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. To find the existing attributes for the managed object, run the okv managed-object attribute list command.
		You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax.
		Attributes that you can enter are as follows:
		• name includes the following:
		– value is the name value.
		– type is either text or uri. The default value is text.
		• contactInfo
		• The following date and time attributes:
		– activationDate
		– deactivationDate
		– protectStopDate
		– processStartDate
		• extractable can be set as follows:
		– TRUE Allows the object to be extracted from Oracle Key Vault.
		– FALSE Prevents the key material within the object from being extracted from Oracle Key Vault. However, the metadata of the object (including object attributes, state, and so on) can still be retrieved from Oracle Key Vault.
		If you do not set the extractable attribute, then this value is inherited from the endpoint's configuration. In the command line, you can only specify the extractable attribute setting that is stricter than the endpoint's effective setting. For example, you cannot set the extractable attribute to TRUE if it is set to FALSE in the endpoint. However, you can always set the extractable attribute value to FALSE in the command line.
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "now" --starts immediately`
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		`"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this date and time`

Parameter/Template Parameter	Required?	Description

		To display the time in UTC format, use the Linux `date` command. For example:
		
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

`--name` / `name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`
		
		• `text`

`--custom-attribute` / `customAttribute`	Optional	Specifies custom defined attribute on security object.
		
		`2017-04-29 18:14:51"}'`
		`--custom-attribute '[ { "name": "x-OKV Certificate Expiration Date",`
		`"value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"} ] '`
		
		`Support simplified data format, name attribute(single instance), in command line --name KEY1`
		
		`Support simplified data format, custom attribute(multi instance),in commandline`
		`--custom-attribute "x-local-name:HR"`
		`--custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		If the type is Byte String, the value must be a valid hexadecimal representation.

Parameter/Template Parameter	Required?	Description
`--crypto-parameter` / `cryptoParameters`	Optional	Includes cryptographic parameters such as Block Cipher Mode, Padding Method, Hashing Algorithm, and Key Role Type. `--crypto-parameter '{"block-cipher-mode":"CBC","padding-method":"NONE","hashing-algorithm":"MD2","key-role-type":"BDK"}'`
`--deactivation-date` / `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`.

`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format.
		
		`"activationDate" : "now" --starts immediately`
		`"activationDate" : "now+PT10M" --starts 10 minutes from now`
		`"activationDate" : "2021-12-20 10:30:00" --starts at this date and time`
		`"activationDate" : "2021-12-20 10:30:00+PT10M" --starts 10 minutes after this date and time`

`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

Example without using JSON

```
okv managed-object private-key register --algorithm RSA --length 2048
         --mask"ENCRYPT" --object /Users/dopark/test/id_rsa
         --name private_0701--activation-date now --deactivation-date
"2030-10-10 10:10:10"
```

JSON Example

1. Generate JSON input for the `okv managed-object private-key register` command:

```
okv managed-object private-key register --generate-json-input
```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `reg_private_key.json`) and then edit it to specify the appropriate private key settings:

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "private-key",
    "action" : "register",
    "options" : {
      "object" : "./priv_key.pem",
      "algorithm" : "RSA",
      "length" : "2048",
      "mask" : [ "ENCRYPT", "DECRYPT" ],
      "wallet" : "hr_wallet",
      "attributes" : {
        "name" : {
          "value" : "CERT-APPID-103",
          "type" : "text"
        },
        "contactInfo" : "psmith@example.com",
        "activationDate" : "2020-12-31 09:00:00",
        "deactivationDate" : "2024-12-31 09:00:00",
        "processStartDate" : "2020-12-31 09:00:00",
        "protectStopDate" : "2024-12-31 09:00:00"
      }
    }
  }
}
```

3. Run the `okv managed-object private-key register` command using the generated JSON file:

```
okv managed-object private-key register --from-json reg_private_key.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "uuid" : "2F9E2A31-D15A-4F5B-BFA0-761892021DBE"
  }
}
```

Example Using Output Format Text

```
okv managed-object private-key register --output_format text --object
private_key_file_path --algorithm cryptographic_algorithm --length key_length --mask
cryptographic_usage_mask --wallet wallet_name
```

Output

Output similar to the following appears:

```
2F9E2A31-D15A-4F5B-BFA0-761892021DBE
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object public-key get

The `okv managed-object public-key get` command retrieves a public key.

Required Authorization

The endpoint must have read permission on the public key.

Syntax

```
okv managed-object public-key get  --output_format text|json --uuid UUID [--wrap-key-uuid wrapKeyUUID]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "public-key",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE",
      "wrapKeyUUID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the public key. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--wrap-key-uuid` / `wrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for wrapping. When specified, the managed object will be retrieved from the KMIP server in a wrapped format using the given key. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are 128, 256.

Parameter/Template Parameter	Required?	Description
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object public-key get` command:

   ```
   okv managed-object public-key get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_public_key.json`) and then edit it to specify the UUID of the public key:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "public-key",
       "action" : "get",
       "options" : {
         "uuid" : "11652909-D019-4F3B-BFB9-791723095005"
       }
     }
   }
   ```

3. Run the `okv managed-object public-key get` command using the generated JSON file:

   ```
   okv managed-object public-key get  --from-json get_public_key.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "object" : -----BEGIN PUBLIC KEY-----
   MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA7DIA9SO/D+YEIweRLGHg
   +clQVoeYtogpm60GMo1vjif7tLzIhR/8QamDDSseq9yB8R7nWoqNi98g+VypNv6S
   3rU8cp3BdW+187xrU47c03xtKgiHZ16RwxoxrN3rJarw2UcrJAk/zUR1iw4bOZqK
   3aKQOYaFIRd2niQApjixPSwhnLU6aEzRkew92lVfN4qafeaAeMbEqsokB7++vD0b
   hryJw9RXSkMmObE1S1sxCjK4eur452N0jHHwc2jdsCoKCpnSVDHPBKkDCXkMBtQy
   XAUC/q8MFiXKM6IVsqnKOPDzC0nhUsDDiO4P7ZXPK8th3L9V5v+DLbOXEOJoNRdK
   uQIDAQAB
   -----END PUBLIC KEY-----
   ```

```
    }
  }
```

Example Using Output Format Text

```
okv managed-object public-key get --output_format text --uuid 11652909-D019-4F3B-
BFB9-791723095005
```

Output

Output similar to the following appears:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtK4YrT6A/4tVnadRg0ZT
prsdUwXrIdoqf1+ye/yVkN6RmtR7mthn6WIIrbTVX5MuAkLc6yyuMEc+nLDPZzrU
FXkCAQeVR7sT/hQo74dQHebIfJxgx+uZrlzOgT4Il1qfmjR6y81RjTvAU8ZPdzPb
uXKHZErZVQdoXUw5uFrTNzOegLbYJFI2dZnf3erB7Ho64DckFRoFP05cc3A0iLrL
tzE8CcjAlBlXTGJD4kAtTEet/0TkvuHzBHr23zkfj0kWV3PHGYYC3O+/UzXg/nal
3iTK5yRDkln45AyI/PkfzAFiZ/kX9C66H0WRMxgfaOn/uRNbikFOFK6IPOGcT+0S
/QIDAQAB
-----END PUBLIC KEY-----
```

okv managed-object public-key register

The `okv managed-object public-key register` command registers a public key. You can use this command to also register an SSH public key.

Required Authorization

None

Syntax

```
okv managed-object public-key register [--activation-date activation date] [--algorithm
algorithm] [--crypto-parameter crypto parameter] [--custom-attribute custom attribute]
[--deactivation-date deactivation date] --length length [--mask mask] [--name name] --
object object [--unwrap-key-uuid unwrapKeyUUID] [--private-key-uuid private-key-uuid] [--
ssh-user ssh-user] [--wallet wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "public-key",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "algorithm" : "#RSA",
      "unwrapKeyUUID" : "#VALUE",
      "length" : "#1024,2048,3072,4096(RSA)",
      "sshUser" : "#VALUE",
      "mask" : [ "#VERIFY", "#ENCRYPT", "#DECRYPT","#WRAP_KEY", "#UNWRAP_KEY",
                 "#EXPORT", "#DERIVE_KEY", "#TRANSLATE_ENCRYPT",
                 "#TRANSLATE_DECRYPT", "#TRANSLATE_WRAP", "#TRANSLATE_UNWRAP" ],
      "privateKeyUUID" : "#VALUE",
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
```

```
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ],
      "cryptoParameters" : [ {
        "blockCipherMode" : "#VALUE",
        "paddingMethod" : "#VALUE",
        "hashingAlgorithm" : "#VALUE",
        "keyRoleType" : "#VALUE"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Optional	File path to the public-key object. If no file path is provided, a prompt to supply the object in the command line will be displayed.
`--algorithm` / `algorithm`	Optional	Cryptographic algorithm. The default value is RSA.
`--unwrap-key-uuid` / `unwrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for unwrapping. When specified, the provided object is expected to be in a wrapped form. It will be unwrapped in the KMIP server using the specified key and will then be registered. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are `128, 256`.
`--length` / `length`	Required	Key length for the algorithm. Choose from the following values: • `1024` • `2048` • `4096`

Parameter/Template Parameter	Required?	Description
`--mask` / `mask`	Optional	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_WRAP` • `TRANSLATE_UNWRAP` • `WRAP_KEY` • `UNWRAP_KEY` • `VERIFY` The default values are: • `UNWRAP_KEY` • `ENCRYPT` • `VERIFY`
`----private-key-uuid` / `privateKeyUUID`	Optional	Universally unique ID (UUID) of the private key associated with the public key being registered. To find the unique identifier for the key, run the `okv manage-access wallet list-objects` command or the `okv admin endpoint list-objects` command.
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.

Parameter/Template Parameter	Required?	Description

`/attributes`	Optional	Attribute names and their values. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command.
		You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax.
		Attributes that you can enter are as follows:
		• `name` includes the following:
		– `value` is the name value.
		– `type` is either `text` or `uri`.
		• `contactInfo`
		• The following date and time attributes:
		– `activationDate`
		– `deactivationDate`
		– `protectStopDate`
		– `processStartDate`
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "now"` *`--starts immediately`*
		`"activationDate" : "now+PT10M"` *`--starts 10 minutes from now`*
		`"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		To display the time in UTC format, use the Linux `date` command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

Parameter/Template Parameter	Required?	Description

`--name` / `name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

`--custom-attribute` / `customAttribute`	Optional	Specifies custom defined attribute on security object.
		
		`2017-04-29 18:14:51"}'`
		`--custom-attribute '[ { "name": "x-OKV Certificate Expiration Date", "value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"} ] '`
		
		`Support simplified data format, name attribute(single instance), in command line --name KEY1`
		
		`Support simplified data format, custom attribute(multi instance),in commandline`
		`--custom-attribute "x-local-name:HR"`
		`--custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		If the type is Byte String, the value must be a valid hexadecimal representation.

`--crypto-parameter` / `cryptoParameters`	Optional	Includes cryptographic parameters such as Block Cipher Mode, Padding Method, Hashing Algorithm, and Key Role Type. `--crypto-parameter '{"block-cipher-mode":"CBC","padding-method":"NONE","hashing-algorithm":"MD2","key-role-type":"BDK"}'`

Parameter/Template Parameter	Required?	Description
`--deactivation-date` / `deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`.
`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It has the following format. `"activationDate" : "now"` *`--starts immediately`* `"activationDate" : "now+PT10M" --starts 10 minutes from now` `"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`* `"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.
`--ssh-user` /`sshUser`	Optional	SSH user name. The SSH user is intended to track the actual consumer of the SSH keys, a human, an application, or a machine.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object public-key register` command:

   ```
   okv managed-object public-key register --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `reg_public_key.json`) and then edit it to specify the appropriate public key settings:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "public-key",
       "action" : "register",
       "options" : {
         "object" : "./key.pub",
   ```

```
        "algorithm" : "RSA",
        "length" : "2048",
        "mask" : [ "ENCRYPT", "DECRYPT" ],
        "privateKeyUUID" : "2F9E2A31-D15A-4F5B-BFA0-761892021DBE ",
        "wallet" : "hr_wallet",
        "attributes" : {
          "name" : {
            "value" : " FINDB-PROD-PUBKEY ",
            "type" : "text"
          },
          "contactInfo" : "psmith@example.com",
          "activationDate" : "2020-12-31 09:00:00",
          "deactivationDate" : "2024-12-31 09:00:00",
          "processStartDate" : "2020-12-31 09:00:00",
          "protectStopDate" : "2024-12-31 09:00:00"
        }
      }
    }
  }
```

3. Run the `okv managed-object public-key register` command using the generated JSON file:

   ```
   okv managed-object public-key register --from-json reg_public_key.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "uuid" : "11652909-D019-4F3B-BFB9-791723095005"
     }
   }
   ```

Example without using JSON

```
okv managed-object public-key register   --object
id_rsa.pub  --algorithm RSA ---length 2048 -mask "ENCRYPT"
--private-key-uuid  95092BD2-B546-4F9A-BF0B-D8ECDC548546 --name hr_public_key
--activation-date now --deactivation-date "2030-10-10 10:10:10
```

Example Using Output Format Text

```
okv managed-object public-key register --output_format text --object
public_key_file_path --algorithm cryptographic_algorithm --length key_length --mask
cryptographic_usage_mask --private-key-uuid private_key_uuid --wallet wallet_name
```

Output

Output similar to the following appears:

```
11652909-D019-4F3B-BFB9-791723095005
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object secret get

The `okv managed-object secret get` command retrieves the secret data from a security object of type secret.

Required Authorization

The endpoint must have read permission on the secret object.

Syntax

```
okv managed-object secret get --output_format text|json --uuid UUID [--wrap-key-uuid wrapKeyUUID]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "secret",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE",
      "wrapKeyUUID" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/ Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the security object. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--wrap-key-uuid` / `wrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for wrapping. When specified, the managed object will be retrieved from the KMIP server in a wrapped format using the given key. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are `128, 256`.
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object secret get` command:

   ```
   okv managed-object secret get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `secret_get.json`) and then edit it to locate the secret object:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "secret",
       "action" : "get",
       "options" : {
         "uuid" : "D69D2F32-2DBB-4FF3-BF52-95487526E6EC"
       }
     }
   }
   ```

3. Run the `okv managed-object secret get` command using the generated JSON file:

   ```
   okv managed-object secret get --from-json secret_get.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
       "object": "ki3j&8slo73y2ls"
     }
   }
   ```

Example Using Output Format Text

```
okv managed-object secret get --output_format text --uuid D69D2F32-2DBB-4FF3-
BF52-95487526E6EC
```

Output

Output similar to the following appears:

```
ki3j&8slo73y2ls
```

okv managed-object secret register

The `okv managed-object secret register` command registers secret data such as passwords or random seeds.

Required Authorization

None

Syntax

```
okv managed-object secret register [--activation-date activation date] [--custom-
attribute custom attribute] [--deactivation-date deactivation date] [--mask mask] [--
name name] --object object [--unwrap-key-uuid unwrapKeyUUID] [--type type] [--wallet
wallet]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "secret",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "unwrapKeyUUID" : "#VALUE",
      "type" : "#PASSWORD|SEED",
      "mask" : [ "#DERIVE_KEY", "#EXPORT" ],
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#NOW|YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#NOW|YYYY-MM-DD HH:mm:ss"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER|BYTE_STRING"
      } ]
    }
  }
}
```

Parameters

Parameter/Template	Required?	Description
`--object` / `object`	Optional	File path to the secret object. If no file path is provided, a prompt to supply the object in the command line will be displayed.

Parameter/Template	Required?	Description
`--unwrap-key-uuid` / `unwrapKeyUUID`	Optional	Universally unique ID (UUID) of the symmetric key used for unwrapping. When specified, the provided object is expected to be in a wrapped form. It will be unwrapped in the KMIP server using the specified key and will then be registered. **Note** Supported wrapping key algorithm is AES. Supported wrapping key lengths are `128, 256`.
`--type` / `type`	Optional	Enter one of the following values: • `SEED` • `PASSWORD` (Default value)
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.
`--mask` / `mask`	Optional	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `EXPORT` • `DERIVE_KEY` (Default value)

Parameter/Template	Required?	Description

`/attributes`	Optional	Attribute names and their values. Enclose this value in double quotation marks if the value contains spaces, slashes, or colons. To find the existing attributes for the managed object, run the `okv managed-object attribute list` command.
		You cannot specify attributes at the command line. If you want to use attributes, then you must use the JSON syntax.
		Attributes that you can enter are as follows:
		• `name` includes the following:
		– `value` is the name value.
		– `type` is either `text` or `uri`. The default value is `text`.
		• `contactInfo`
		• The following date and time attributes:
		– `activationDate`
		– `deactivationDate`
		– `protectStopDate`
		– `processStartDate`
		You can use different ways to set the date and time. Examples are as follows:
		`"activationDate" : "now"` *`--starts immediately`*
		`"activationDate" : "now+PT10M"` *`--starts 10 minutes from now`*
		`"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		To display the time in UTC format, use the Linux `date` command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.

Parameter/Template	Required?	Description

`--name/ name`	Optional	Specifies the name of a security object. The allowed values are :
		• `value`
		• `type`: Default value is `text`.
		
		`-Support complex data format, name attribute in command line`
		`--name '{"value" : "KEY1", "type" : "uri"}'or`
		`--name '{"value" : "KEY1", "type" : "text}'`
		`-Support simplified data format, name attribute in command line. when type is "text" as a default:`
		`--name KEY1`

`--custom-attribute / customAttribute`	Optional	Specifies custom defined attribute on security object.
		
		`2017-04-29 18:14:51"}'`
		`--custom-attribute '[ { "name": "x-OKV Certificate Expiration Date",`
		`"value" : "2017-04-29 18:14:51"}, { "name": "x-local-name", "value" : "HR"} ] '`
		
		`Support simplified data format, name attribute(single instance), in command line --name KEY1`
		
		`Support simplified data format, custom attribute(multi instance),in commandline`
		`--custom-attribute "x-local-name:HR"`
		`--custom-attribute ' [ “x-local-name:HR”,“x-local-id:100” ]'`
		
		If the type is Byte String, the value must be a valid hexadecimal representation.

`--deactivation-date/ deactivationDate`	Optional	Specifies when to deactivate a security object. It has the same format as `activation-date`.

Parameter/Template	Required?	Description
`--activation-date` / `activationDate`	Optional	Specifies when to activate a security object. It will have the following format. `"activationDate" : "now"` *`--starts immediately`* `"activationDate" : "now+PT10M"` *`--starts 10 minutes from now`* `"activationDate" : "2021-12-20 10:30:00"` *`--starts at this date and time`* `"activationDate" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

Example without using JSON

```
okv managed-object secret register --type PASSWORD --mask "DERIVE_KEY"
--name secret_0701 --object /Users/dopark/test/my.secret --activation-date now
--deactivation-date "2030-10-10 10:10:10
```

JSON Example

1. Generate JSON input for the `okv managed-object secret register` command:

   ```
   okv managed-object secret register --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `reg_secret.json`) and then edit it to register the secret object:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "secret",
       "action" : "register",
       "options" : {
         "object" : "./hr_db_connect_password.txt",
         "type" : "PASSWORD",
   ```

```
        "mask" : [ "DERIVE_KEY" ],
        "wallet" : "hr_wallet",
        "attributes" : {
          "name" : {
            "value" : "HR-DB-CONNECT-PASSWORD",
            "type" : "text"
          },
          "contactInfo" : "psmith@example.com",
          "activationDate" : "2020-12-31 09:00:00",
          "deactivationDate" : "2024-12-31 09:00:00",
          "processStartDate" : "2020-12-31 09:00:00",
          "protectStopDate" : "2024-12-31 09:00:00"
        }
      }
    }
  }
```

3. Run the `okv managed-object secret register` command using the generated JSON file:

   ```
   okv managed-object secret register --from-json reg_secret.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
       "uuid": "0F54D31A-ABA0-4F15-BF67-1B7513DD8634"
     }
   }
   ```

Example Using Output Format Text

```
okv managed-object secret register --output_format text --object object_name --type
PASSWORD|SEED --wallet wallet_name --mask cryptogrpahic_usage_mask
```

Output

Output similar to the following appears:

```
0F54D31A-ABA0-4F15-BF67-1B7513DD8634
```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv managed-object wallet add-member

The `okv managed-object wallet add-member` command adds a security object to a wallet as its member.

This command authenticates with the endpoint's client certificate.

Required Authorization

The endpoint must have read-modify permission on the object and manage-wallet access (`MW`) on the wallet.

Syntax

```
okv managed-object wallet add-member -output_format text|json --uuid UUID --wallet
wallet_name
```

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "wallet",
    "action": "add-member",
    "options": {
      "uuid": "#VALUE",
      "wallet": "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the managed object that is being added to the wallet. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object wallet add-member` command:

   ```
   okv managed-object wallet add-member --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `add_wallet_member.json`) and then edit it to add the security object to the wallet:

```
{
  "service": {
    "category": "managed-object",
    "resource": "wallet",
    "action": "add-member",
    "options": {
      "uuid": "D69D2F32-2DBB-4FF3-BF52-95487526E6EC",
      "wallet": "hr_wallet"
    }
  }
}
```

3. Run the `okv managed-object wallet add-member` command using the generated JSON file:

```
okv managed-object wallet add-member --from-json add_wallet_member.json
```

Output similar to the following appears:

```
{
  "result": "Success"
}
```

Example Using Output Format Text

```
okv managed-object wallet add-member --output_format text --uuid UUID --wallet
wallet_name
```

Output

Output similar to the following appears:

- exit code 0 - Indicates Success
- exit code 1- Indicates Failure

okv managed-object wallet delete-member

The `okv managed-object wallet delete-member` command deletes the membership of the managed-object from a wallet.

This command authenticates with the endpoint's client certificate.

Required Authorization

The endpoint must have read-modify permission on the object and manage-wallet access (`MW`) on the wallet.

Syntax

```
okv managed-object wallet delete-member -output_format text|json--uuid UUID --wallet
wallet_name
```

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "wallet",
    "action": "delete-member",
```

```
    "options": {
      "uuid": "#VALUE",
      "wallet": "#VALUE"
    }
  }
}
```

Parameters

Parameter/ Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the managed object in the wallet. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, run the `okv manage-access wallet list` command.
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object wallet delete-member` command:

   ```
   okv managed-object wallet delete-member --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `delete_wallet_member.json`) and then edit it to delete the security object from the wallet:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "wallet",
       "action": "delete-member",
       "options": {
         "uuid": "D69D2F32-2DBB-4FF3-BF52-95487526E6EC",
         "wallet": "hr_wallet"
       }
     }
   }
   ```

3. Run the `okv managed-object wallet delete-member` command using the generated JSON file:

   ```
   okv managed-object wallet delete-member --from-json delete_wallet_member.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success"
   }
   ```

Example Using Output Format Text

```
okv managed-object wallet delete-member --output_format text --uuid UUID --wallet
wallet_name
```

Output

Output similar to the following appears:

- exit code 0 - Indicates Success
- exit code 1- Indicates Failure

okv managed-object wallet list

The `okv managed-object wallet list` command lists wallets that have their access granted to the endpoint used to connect to Oracle Key Vault.

This command authenticates with the endpoint's client certificate.

Required Authorization

None, but this command returns only those wallets to which the current endpoint is granted access.

Syntax

```
okv managed-object wallet list
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "wallet",
    "action" : "list"
  }
}
```

Parameters

None

Parameter/Template Parameter	Required?	Description
`--output_format`	Optional	Specifies output format of the command. The command completes with an exit code 0, when command is executed successfully, and exit code 1, when the command fails and generates a relevant error message. The default value is `text`.

> **Note**
>
> Use the CLI command syntax to specify the output_format option. By default, the output format is JSON. However, the output_format option with a value as 'text' displays the output in text format. Use of text output format removes the need to parse JSON output. The option is useful when the output of a command serves as input for another command.

JSON Example

1. Generate JSON input for the `okv managed-object wallet list` command:

   ```
   okv managed-object wallet list --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `wallet_list.json`).

3. Run the `okv managed-object wallet list` command using the generated JSON file:

   ```
   okv managed-object wallet list --from-json wallet_list.json
   ```

   Output similar to the following appears:

   ```
   {
     "result": "Success",
     "value": {
       "wallets": ["HR_WALLET", "SALES_WALLET"]
     }
   }
   ```

Example Using Output Format Text

```
okv managed-object wallet list -output_format text
```

Output

Output similar to the following appears:

```
"HR_WALLET","SALES_WALLET
```

6

Cryptographic Commands

You can use the cryptographic commands to encrypt and decrypt the provided data using the Oracle Key Vault managed security objects.

- okv crypto data decrypt
  The `okv crypto data decrypt` command performs a decrypt operation on the given ciphertext data using the Oracle Key Vault managed security object that is within the Oracle Key Vault server, and returns the decrypted data.
- okv crypto data encrypt
  The `okv crypto data encrypt` command performs an encrypt operation on the given plaintext data using the Oracle Key Vault managed security object that is within the Oracle Key Vault server, and returns the encrypted data.
- okv crypto data sign
  The `okv crypto data sign` command performs a sign operation on the given message file using the Oracle Key Vault managed security object that is within the Oracle Key Vault server, and returns the signature data.
- okv crypto data sign-verify
  The `okv crypto data sign-verify` command performs a signature verification operation on the given signature data and message file using the Oracle Key Vault managed security object that is within the Oracle Key Vault server, and returns whether or not the signature is valid.

okv crypto data decrypt

The `okv crypto data decrypt` command performs a decrypt operation on the given ciphertext data using the Oracle Key Vault managed security object that is within the Oracle Key Vault server, and returns the decrypted data.

Required Authorization

The endpoint must have read permission on the key used for the decryption.

Syntax

```
okv crypto data decrypt --uuid UUID --data file_path
--block-cipher-mode block_cipher_mode --padding padding --iv file_path
--authenticated-encryption-additional-data file_path
--authenticated-encryption-tag file_path --data-format data_format
--decrypted-data output_file_path
```

JSON Input File Template

```
{
  "service": {
    "category": "crypto",
    "resource": "data",
    "action": "decrypt",
    "options": {
      "uuid": "#VALUE",
```

```
      "data" : "#VALUE",
      "blockCipherMode" : "#CBC|ECB|CFB|OFB|GCM",
      "padding" : "#NONE|ZEROS|PKCS5",
      "iv" : "#VALUE",
      "authenticatedEncryptionAdditionalData" : "#VALUE",
      "authenticatedEncryptionTag" : "#VALUE",
      "dataFormat": "#HEX|BASE64",
      "decryptedData": "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the key to use for the decryption. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--data` / `data`	Required	File path to the ciphertext data that needs to be decrypted
`--block-cipher-mode` / `blockCipherMode`	Optional	Block Cipher Mode. Values are as follows: • `CBC` for cipher block chaining • `CFB` for cipher feedback • `ECB` for electronic codebook • `GCM` for Galois/counter • `OFB` for output feedback If you omit this setting, then Oracle Key Vault uses the cryptographic parameters that are associated with the key.
`--padding` / `padding`	Optional	Padding. Values are as follows: • `NONE` • `ZEROS` • `PKCS5` If you omit this setting, then Oracle Key Vault uses the cryptographic parameters that are associated with the key.
`--iv` / `iv`	Optional	File path of the initialization vector (IV) to use for the decrypt operation. You must use the same initialization vector that was used during encryption.
`--authenticated-encryption-additional-data` / `authenticatedEncryptionAdditionalData`	Optional	File path of authenticated encryption additional data to use for the decrypt operation. You must specify the same authenticated encryption additional data that was used during encryption.
`--authenticated-encryption-tag` / `authenticatedEncryptionTag`	Optional	File path of the authenticated encryption tag to use for the decrypt operation. You must specify the same authenticated encryption tag that was generated during encryption.

Parameter/Template Parameter	Required?	Description
`--data-format` / `dataFormat`	Optional	Data format. Format of the data in input and output files. If not specified, data is read and written as binary data. Values are as follows: • `HEX` • `BASE64`
`--decrypted-data` / `decryptedData`	Required	File path where the decrypted data is written. If the provided output file does not exist, then an error results. If the file is present, then it is overwritten with the decrypted data.

JSON Example

1. Generate JSON input for the `okv crypto data decrypt` command:

   ```
   okv crypto data decrypt --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `key_decrypt.json`) and then edit it to include the decryption settings that you want. For example:

   ```
   {
     "service": {
       "category": "crypto",
       "resource": "data",
       "action": "decrypt",
       "options": {
         "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
         "data" : "/okv/opt/data",
         "blockCipherMode" : "GCM",
         "padding" : "ZEROS",
         "iv" : "/okv/opt/iv",
         "authenticatedEncryptionAdditionalData" : "/okv/opt/keys/
   authenticatedEncryptionAdditionalData",
         "authenticatedEncryptionTag" : "/okv/opt/keys/authenticatedEncryptionTag",
         "dataFormat": "HEX",
         "decryptedData": "/okv/opt/keys/decrypted_data"
       }
     }
   }
   ```

3. Run the `okv crypto data decrypt` command using the generated JSON file:

   ```
   okv crypto data decrypt --from-json key_decrypt.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "decryptedData" : "/okv/opt/keys/decrypted_data"
     }
   }
   ```

okv crypto data encrypt

The `okv crypto data encrypt` command performs an encrypt operation on the given plaintext data using the Oracle Key Vault managed security object that is within the Oracle Key Vault server, and returns the encrypted data.

Required Authorization

The endpoint must have read permission on the key used for the encryption.

Syntax

```
okv crypto data encrypt --uuid UUID --data file_path
--block-cipher-mode block_cipher_mode --padding padding
--random-iv random_iv --iv file_path
--authenticated-encryption-additional-data file_path --data-format data_format
--encrypted-data output_file_path --iv-out output_file_path
--authenticated-encryption-tag output_file_path
```

JSON Input File Template

```
{
  "service": {
    "category": "crypto",
    "resource": "data",
    "action": "encrypt",
    "options": {
      "uuid": "#VALUE",
      "data" : "#VALUE",
      "blockCipherMode" : "#CBC|ECB|CFB|OFB|GCM",
      "padding" : "#NONE|ZEROS|PKCS5",
      "randomIV" : "#TRUE|FALSE",
      "iv" : "#VALUE",
      "authenticatedEncryptionAdditionalData" : "#VALUE",
      "dataFormat": "#HEX|BASE64",
      "encryptedData": "#VALUE",
      "ivOut": "#VALUE",
      "authenticatedEncryptionTag" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the key to use for the encryption. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--data` / `data`	Required	File path to the plaintext data that needs to be encrypted

Parameter/Template Parameter	Required?	Description
`--block-cipher-mode` / `blockCipherMode`	Optional	Block Cipher Mode. Values are as follows: • `CBC` for cipher block chaining • `CFB` for cipher feedback • `ECB` for electronic codebook • `GCM` for Galois/counter • `OFB` for output feedback If you omit this setting, then Oracle Key Vault uses the cryptographic parameters that are associated with the key. **Note** If you are using the `block-cipher-mode` as `GCM`, then set the `--authenticated-encryption-tag` with the parameter.
`--padding` / `padding`	Optional	Padding. Values are as follows: • `NONE` • `ZEROS` • `PKCS5` If you omit this setting, then Oracle Key Vault uses the cryptographic parameters that are associated with the key.
`--random-iv` / `randomIV`	Optional	Indicates whether the Oracle Key Vault server should use random initialization vector (IV). Values are as follows: • `TRUE` • `FALSE` (default) Oracle Key Vault uses the `randomIV` value only when the IV is not provided in the input for this command. If you omit the IV and if `randomIV` is set to `TRUE` in the input, then Oracle Key Vault uses a random IV for the encrypt operation.
`--iv` / `iv`	Optional	File path of the IV to use for the encrypt operation. If you include the IV file path in the `okv crypto data encrypt` command, and when you run `okv crypto data decrypt`, use this IV file path.
`--authenticated-encryption-additional-data` / `authenticatedEncryptionAdditionalData`	Optional	File path of authenticated encryption additional data to use for the encrypt operation. If you include the authenticated encryption additional data file path in the `okv crypto data encrypt` command, and when you run `okv crypto data decrypt`, use this authenticated encryption file path.

Parameter/Template Parameter	Required?	Description
`--data-format` / `dataFormat`	Optional	Data format. Format of the data in input and output files. If not specified, data is read and written as binary data. Values are as follows: • `HEX` • `BASE64`
`--encrypted-data` / `encryptedData`	Required	File path where the encrypted data is written. If the provided output file does not exist, then an error results. If the file is present, then it is overwritten with the encrypted data.
`--iv-out` / `ivOut`	Optional	File path where the response IV is written. If the provided output file does not exist, then an error results. If the file is present, then it is overwritten with the response IV. The IV is returned in `ivOut` only when `iv` is not provided and `randomIV` is set to true in the input. If `iv` is provided in the input, then Oracle Key Vault ignores the `ivOut` parameter. If you include the response IV file path in the `okv crypto data encrypt` command, and when you run `okv crypto data decrypt`, use this response IV file path.
`--authenticated-encryption-tag` / `authenticatedEncryptionTag`	Optional	File path where the response authenticated encryption tag is written. If the provided output file does not exist, then an error results. If the file is present, then it is overwritten with the response authenticated encryption tag. The authenticated encryption tag that is returned should be used for decrypting the cipher text. If you include the response authenticated encryption tag file path in the `okv crypto data encrypt` command, and when you run `okv crypto data decrypt`, use this response authenticated encryption tag file path.

JSON Example

1. Generate JSON input for the `okv crypto data encrypt` command:

   ```
   okv crypto data encrypt --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `key_encrypt.json`) and then edit it to include the encryption settings that you want. ***Keep a record of the values that you use during encryption along with the generated ivOut and authenticatedEncryptionTag, if any. You must provide the same values when decrypting the ciphertext.*** For example:

   ```
   {
     "service": {
       "category": "crypto",
       "resource": "data",
       "action": "encrypt",
       "options": {
         "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
         "data" : "/okv/opt/data",
   ```

```
      "blockCipherMode" : "GCM",
      "padding" : "ZEROS",
      "iv" : "/okv/opt/iv",
      "authenticatedEncryptionAdditionalData" : "/okv/opt/keys/
authenticatedEncryptionAdditionalData",
      "dataFormat": "HEX",
      "encryptedData": "/okv/opt/keys/encrypted_data",
      "authenticatedEncryptionTag" : "/okv/opt/keys/authenticatedEncryptionTag"
    }
  }
}
```

3. Run the `okv crypto data encrypt` command using the generated JSON file:

```
okv crypto data encrypt --from-json key_encrypt.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "encryptedData" : "/okv/opt/keys/encrypted_data",
    "authenticatedEncryptionTag" : "/okv/opt/keys/authenticatedEncryptionTag"
  }
}
```

> **Note**
>
> Oracle Key Vault can encrypt maximum 32KB of data.

okv crypto data sign

The `okv crypto data sign` command performs a sign operation on the given message file using the Oracle Key Vault managed security object that is within the Oracle Key Vault server, and returns the signature data.

Required Authorization

The endpoint must have read permission on the private key used for signing.

Syntax

```
okv crypto data sign --uuid <UUID> --message-file <filePath> --message <message> --
message-type <messageType> --digital-signature-algorithm <digitalSignatureAlgoritm> --
signature-data <output file path> --output_format TEXT
```

JSON Input File Template

```
{
  "service": {
    "category": "crypto",
    "resource": "data",
    "action": "sign",
    "options": {
      "uuid": "#VALUE",
      "message" : "#VALUE",
      "messageFile" : "#VALUE",
      "messageType" : "#RAW|DIGEST",
      "digitalSignatureAlgorithm" : "#RSASSA_PKCS1_v1_5_SHA256|RSASSA_PKCS1_v1_5_SHA384|
```

```
RSASSA_PKCS1_v1_5_SHA512|RSASSA_PSS_SHA256|RSASSA_PSS_SHA384|RSASSA_PSS_SHA512",
      "signatureData": "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Unique Identifier of the key to be used for the signature operation. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.
`--message/message`	Optional	The data that needs to be signed. • If message type is DIGEST, it should be HEX. • If message type is RAW, it can be any type. • Either `--message/message` or `--message-file/messageFile` options are required.
`--message-file/ messageFile`	Optional	Denotes the file path to the data that needs to be signed. Specified by the `message-file` or `messageFile` argument. Either `--message/message` or `--message-file/messageFile` options is required
`--message-type/ messageType`	Optional	Denotes the type of data specified by the `message` or `message-file/messageFile` argument - `RAW` or `DIGEST`. The Default value is `RAW`
`--digital-signature-algorithm` / `digitalSignatureAlgorithm`	Optional	Digital Signature Algorithm Supported algorithms are • * `RSASSA_PKCS1_v1_5_SHA256` • * `RSASSA_PKCS1_v1_5_SHA384` • * `RSASSA_PKCS1_v1_5_SHA512` • * `RSASSA_PSS_SHA256` • * `RSASSA_PSS_SHA384` • * `RSASSA_PSS_SHA512` Default: `RSASSA_PKCS1_v1_5_SHA256`

Parameter/Template Parameter	Required?	Description

`--signature-data` / / `signatureData`	Optional	Path to the file where the received signed data should be written to.
		If this option is provided, the output will be:
		`{ "result" : "Success" }`
		If this option is not provided, the output will be:
		`{ "result" : "Success", "value" : { "signatureData" : "3258D33DFB12F97....86419A35F32BA903ADDEB3" } }`

JSON Example

1. Generate JSON input for the `okv crypto data sign` command:

   ```
   okv crypto data sign --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `key_sign.json`) and then edit it to include the sign settings that you want. For example:

   ```
   {
    "service": {
      "category": "crypto",
      "resource": "data",
      "action": "sign",
      "options": {
        "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
        "message" : "Example message to sign",
        "messageType" : "RAW",
        "digitalSignatureAlgorithm" : "RSASSA_PKCS1_v1_5_SHA256"
      }
    }
   }
   ```

3. Run the `okv crypto data sign` command using the generated JSON file:

   ```
   okv crypto data sign --from-json key_sign.json
   ```

   If "--signature-data/signatureData" is provided, the output will be:

   ```
   {
     "result" : "Success",
     "value" : {
       "signatureData" :
   ```

```
"12634F979551C19ADAEB69733853ADB41405FF108E479393AF8B82140186F7244A41F7E36BA1129E6745
3B36297BB91115C4B10B02101AA8068E251B74B7374E975E1E9C1EEACDCB73BAACF4E05359563A8806B49
AA9263ECF61A0D4A0769F1CA5C3CEC0B0B8B4F4F470C5E78F01549C04A491CE346916ECC55E5AA6E2EEA4
2A3909A38A8090C341FAFEE7C1547D7BC4509CDC65728729011F4301DFB105CF2A0F6B1799D4B9B296677
89E6EA1A4319D14E7B92BBC2E68F3DB20CA8B8270FC20C272F638202F3D68248B7AF12750C2A22DF15988
6AC2456DBAA4CC94A90A064D771106619C103DCCC66C0815FA9FF3349A03E0E3D9696984E6A826EAA507C
32F"
  }
}
```

okv crypto data sign-verify

The `okv crypto data sign-verify` command performs a signature verification operation on the given signature data and message file using the Oracle Key Vault managed security object that is within the Oracle Key Vault server, and returns whether or not the signature is valid.

Required Authorization

The endpoint must have read permission on the public key used for signature verification.

Syntax

```
okv crypto data sign-verify --uuid <UUID> --message-file <filePath> --message <message>
--signature-data <filePath> --digital-signature-algorithm <digitalSignAlgoritm> --
text_output TEXT
```

JSON Input File Template

```
{
  "service": {
    "category": "crypto",
    "resource": "data",
    "action": "sign-verify",
    "options": {
      "uuid": "#VALUE",
      "message" : "#VALUE",
      "messageFile" : "#VALUE",
      "messageType" : "#RAW|DIGEST",
      "signatureData" : "#VALUE",
      "digitalSignatureAlgorithm" : "#RSASSA_PKCS1_v1_5_SHA256|RSASSA_PKCS1_v1_5_SHA384|
RSASSA_PKCS1_v1_5_SHA512|RSASSA_PSS_SHA256|RSASSA_PSS_SHA384|RSASSA_PSS_SHA512"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the key to use for the signature verify. To find the unique identifier for the object, in the Oracle Key Vault management console, click the **Keys & Wallets** tab, and then click **Keys & Secrets** in the left navigation window. In the Keys & Secrets table, check the Unique Identifier column.

Parameter/Template Parameter	Required?	Description
`--message` / `message`	Optional	The data that is passed to the signing operation (for the algorithms requiring the original data to verify a signature). Either `--message/message` or `--message-file/messageFile` options is required
`--message-file` / `--messageFile`	Optional	Specifies the file path to the data that is passed to the signing operation (for the algorithms requiring the original data to verify a signature). Denotes the file path to the data that needs to be signed. Specified by the `message-file` or `messageFile` argument.
`--message-type` / `messageType`	Optional	Denotes the type of data specified by the `message` or `message-file/messageFile` argument - `RAW` or `DIGEST`.
`--signature-data` / `signatureData`	Required	Denotes the file path to the signature that needs to be verified.
`--digital-signature-algorithm` / `digitalSignatureAlgorithm`	Optional	Digital Signature of the algorithm.
`--output_format`	Optional	Provides the output format.

JSON Example

1. Generate JSON input for the `okv crypto data sign-verify` command:

   ```
   okv crypto data sign-verify --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `key_sign-verify.json`) and then edit it to include the decryption settings that you want. For example:

   ```
   {
     "service": {
       "category": "crypto",
       "resource": "data",
       "action": "sign-verify",
       "options": {
         "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A",
         "message" : "Example message to sign",
         "messageType" : "RAW",
         "signatureData": "/tmp/signature_data",
         "digitalSignatureAlgorithm" : "RSASSA_PKCS1_v1_5_SHA256"
       }
     }
   }
   ```

3. Run the `okv crypto data sign-verify` command using the generated JSON file:

   ```
   okv crypto data sign-verify --from-json key_sign-verify.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
   ```

```
  "value" : {
    "validityIndicator" : "VALID"
  }
}
```

Example

The following example shows how to use openssl to verify signature generated by `okvutil sign` command.

1. openssl does not support verification of signatures in HEX format. As the signature generated by okvutil sign command is in HEX format therefore the signature can be converted into binary format using xxd or some other utility.

   ```
   xxd -r -p /tmp/signature_data >
         /tmp/signature_data.bin
   ```

2. Verify signature of a message stored in file message.txt, using signature in binary format stored in /tmp/signature_data.bin and public key stored in file key.pub.

   ```
   openssl dgst -sha256 -verify key.pub -signature /tmp/signature_data.bin
   message.txt
   Verified OK
   ```

7

Monitoring Commands

You can use the monitoring commands to check the Oracle Key Vault configuration, health, and deployment modes.

- okv cluster info get
  The `okv cluster info get` command retrieves status information about a cluster or a cluster node.
- okv cluster status get
  The `okv cluster status get` command retrieves dynamic information about the cluster or the specified cluster node.
- okv metrics application get
  The `okv metrics application get` command retrieves the Oracle Key Vault server application metrics information.
- okv metrics server get
  The `okv metrics server get` command retrieves System Metrics information of an Oracle Key Vault server.
- okv primary-standby info get
  The `okv primary-standby info get` command displays static information about the Oracle Key Vault primary-standby configuration.
- okv primary-standby status get
  The `okv primary-standby status get` command retrieves dynamic information about the Oracle Key Vault primary-standby configuration.
- okv server info get
  The `okv server info get` command retrieves static information about an Oracle Key Vault server.
- okv server status get
  The `okv server status get` command retrieves status information about an Oracle Key Vault server.

okv cluster info get

The `okv cluster info get` command retrieves status information about a cluster or a cluster node.

`okv cluster info get` retrieves the following information:

- Cluster name
- Cluster version
- **Maximum Disable Node Duration** setting
- List of cluster subgroups
- Information of cluster nodes including their configuration mode, status, subgroup, version information, and so on.

Required Authorization

System Administrator role or Monitor privilege. A regular Oracle Key Vault user can also run the command to check the Oracle Key Vault version.

> **Note**
>
> For regular Oracle Key Vault user the command displays the version number only.

Syntax

```
okv cluster info get --node node_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "info",
    "action" : "get",
    "options" : {
      "node" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--node` / `node`	Optional	Name of the node within the current cluster. If you omit this setting, then information for the entire cluster is retrieved.

JSON Example

1. Generate JSON input for the `okv cluster info get` command:

   ```
   okv cluster info get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_cluster_info.json`).
   Depending on the kind of information that you want to find, do one of the following:

   - **Get cluster information for a specific node:** Edit the file to specify the `node` value. For example:

     ```
     {
       "service" : {
         "category" : "cluster",
         "resource" : "info",
         "action" : "get",
         "options" : {
           "node" : "node_1"
         }
     ```

```
  }
}
```

- **Get cluster information for all nodes in the cluster:** Edit the file to remove the node entry:

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "info",
    "action" : "get"
  }
}
```

3. Run the `okv cluster info get` command using the generated JSON file:

```
okv cluster info get --from-json get_cluster_info.json
```

Depending on how you handled the file in Step 2, output similar to the following appears.

- **Get cluster information for a specific node:**

```
{
  "result" : "Success",
  "value" : {
    "clusterSubgroup" : "okvslgrp1",
    "disableDate" : "",
    "installDate" : "2024-10-03 04:05:14",
    "ipAddress" : "100.70.126.1",
    "joinDate" : "2024-10-09 13:20:01",
    "mode" : "Read-Write",
    "nodeID" : "2",
    "nodeName" : "okvnode143",
    "readWritePeer" : "okv020017040e3c",
    "status" : "ACTIVE",
    "version" : "21.10.0.0.0"
  }
}
```

- **Get cluster information for all nodes in the cluster:**

```
{
  "result" : "Success",
  "value" : {
    "clusterName" : "okvcluster1",
    "clusterSubgroups" : [ "okvslgrp1" ],
    "clusterVersion" : "21.10.0.0.0",
    "maximumDisableNodeDuration" : "24 hrs",
    "nodes" : [ {
      "nodeName" : "okv020017040e3c",
      "nodeID" : "1",
      "ipAddress" : "100.70.1.1",
      "mode" : "Read-Write",
      "status" : "ACTIVE",
      "readWritePeer" : "okvnode143",
      "clusterSubgroup" : "okvslgrp1",
      "joinDate" : "2024-10-09 13:13:23",
      "disableDate" : "",
      "version" : "21.10.0.0.0",
      "installDate" : "2024-10-03 10:31:33"
    }, {
      "nodeName" : "okvnode143",
      "nodeID" : "2",
      "ipAddress" : "100.70.1.2",
      "mode" : "Read-Write",
```

```
        "status" : "ACTIVE",
        "readWritePeer" : "okv020017040e3c",
        "clusterSubgroup" : "okvslgrp1",
        "joinDate" : "2024-10-09 13:20:01",
        "disableDate" : "",
        "version" : "21.10.0.0.0",
        "installDate" : "2024-10-03 04:05:14"
      } ]
    }
  }
```

okv cluster status get

The `okv cluster status get` command retrieves dynamic information about the cluster or the specified cluster node.

`okv cluster status get` retrieves the following information:

- Nodes that are read/write pairs
- Nodes that are read-only pairs
- The number and type of unresolved name conflicts
- Alerts that are related to the cluster

Required Authorization

None. Information returned by the command varies depending upon the user's authorization.

Syntax

`okv cluster status get --node` *`node_name`*

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "status",
    "action" : "get",
    "options" : {
      "node" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--node` / `node`	Optional	Name of the node within the current cluster. If you omit this setting, then information for the entire cluster is retrieved.

JSON Example

1. Generate JSON input for the `okv cluster status get` command:

   ```
   okv cluster status get --generate-json-input
   ```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `get_cluster_status.json`). Depending on the kind of information that you want to find, do one of the following:

   - **Get cluster status for a specific node:** Edit the file to specify the `node` value. For example:

     ```
     {
       "service" : {
         "category" : "cluster",
         "resource" : "status",
         "action" : "get",
         "options" : {
           "node" : "node_1"
         }
       }
     }
     ```

   - **Get the status of the cluster**: Edit the file to remove the node entry:

     ```
     {
       "service" : {
         "category" : "cluster",
         "resource" : "status",
         "action" : "get"
       }
     }
     ```

3. Run the `okv cluster status get` command using the generated JSON file:

   ```
   okv cluster status get --from-json get_cluster_status.json
   ```

   Depending on how you handled the file in Step 2, output similar to the following appears.

   - **Get cluster status for a specific node:**

     ```
     {
       "result" : "Success",
       "value" : {
         "mode" : "Read-Write",
         "nameResolutionTime" : "62.79 seconds",
         "nodeID" : 1,
         "nodeName" : "node1",
         "status" : "ACTIVE"
       }
     }
     ```

   - **Get the status of the cluster:**

     ```
     {
       "result" : "Success",
       "value" : {
         "alertsCount" : "1",
         "clusterServiceStatus" : "Up",
         "nodes" : [ {
           "nodeID" : "1",
           "nodeName" : "node1",
           "mode" : "Read-Write",
           "status" : "ACTIVE",
           "nameResolutionTime" : "62.79 seconds"
         }, {
           "nodeID" : "2",
           "nodeName" : "node2",
     ```

```
        "mode" : "Read-Write",
        "status" : "ACTIVE",
        "nameResolutionTime" : "62.79 sec(s)"
      } ],
      "unresolvedConflicts" : {
        "endpointNameConflicts" : "0",
        "endpointGroupNameConflicts" : "0",
        "userNameConflicts" : "0",
        "userGroupNameConflicts" : "0",
        "walletNameConflicts" : "0",
        "kmipObjectNameConflicts" : "0"
      }
    }
  }
}
```

okv metrics application get

The `okv metrics application get` command retrieves the Oracle Key Vault server application metrics information.

Application metrics details of Oracle Key Vault server:

- Number of KMIP Connections
- Number of RESTFul KMIP Connections

Required Authorization

System Administrator role or Monitor privilege

Syntax

```
 okv metrics application get --start-time start_time --end-time end_time --include
include --interval interval --statistic statistic
```

JSON Input File Template

```
{
  "service" : {
    "category" : "metrics",
    "resource" : "application",
    "action" : "get",
    "options" : {
      "startTime" : "#YYYY-MM-DD hh:mm:ss",
      "endTime" : "#NOW|YYYY-MM-DD hh:mm:ss",
      "interval" : "#VALUE",
      "statistic" : "#MEAN|MAX|MIN|COUNT",
      "include" : [ "ALL" ],
      "limit" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--start-time`	Required	Specifies the start time for which results should be returned. It should be entered as timestamp in "YYYY-MM-DD HH:mm:ss" format. Example: Timestamp : YYYY-MM-DD HH:mm:ss
`--end-time`	Required	The end time for which results should be returned. Example: Timestamp : YYYY-MM-DD HH:mm:ss or NOW.
`--interval`	Required	Specifies the time window used to convert given set of raw data points. Enter this value using the ISO 8601 standard. For example, enter PT10M to specify an interval size of 10 Minutes. **Note** The timestamp of the aggregated data point corresponds to the start of the time window during which raw data points are assessed. For example, for a 10 minutes interval, the timestamp "09:00" corresponds to the 10 minutes time window from 09:00 to 09:10. Example: ISO 8601 Duration
`--statistic`	Required	Specifies the aggregation function applied to the given set of raw data points in an interval. Supported functions are MEAN, MIN, MAX and COUNT. Example: MEAN, MIN, MAX, or COUNT.
`--include`	Required	Specifies the list of system resources that requires data. The supported resources are KMIP. Example: Specify "KMIP" to include KMIP and "ALL" to include all resources.
`--limit`	Optional	The maximum number of data should be returned.

JSON Example

1. Generate JSON input for the `okv metrics application get` command:

   ```
   okv metrics application get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated output to a file (for example, `metric_application_get.json`):

   ```
   {
     "service" : {
   ```

```
    "category" : "metrics",
    "resource" : "application",
    "action" : "get",
    "options" : {
      "startTime" : "2023-04-20 20:05:00",
      "endTime" : "NOW",
      "interval" : "PT10M",
      "statistic" : "COUNT",
      "include" : [ "ALL" ],
      "limit" : "3"
    }
  }
}
```

3. Run the `okv metrics application get` command using the generated JSON file:

```
okv metrics application get --from-json metric_application_get.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "metrics" : [ {
      "sampleTime" : "2023-04-20 20:05:03",
      "kmipConnections" : "15",
      "RESTfulKmipConnections" : "5"
    }, {
      "sampleTime" : "2023-04-20 20:15:04",
      "kmipConnections" : "10",
      "RESTfulKmipConnections" : "7"
    }, {
      "sampleTime" : "2023-04-20 20:25:05",
      "kmipConnections" : "3",
      "RESTfulKmipConnections" : "1"
    } ]
  }
}
```

okv metrics server get

The `okv metrics server get` command retrieves System Metrics information of an Oracle Key Vault server.

Installed resources on an Oracle Key Vault server shown under reserved resources, including:

- Number of CPU Cores
- Total amount of memory installed in KB

> **Note**
>
> There can be multiple entries for reserved resources in case there's a change in resources installed or an Oracle Key Vault server reboot.

Server metrics details of Oracle Key Vault server are divided into four categories with each containing metrics related to the category:

- CPU Usage

  - CPU Utilization Percentage
  - CPU load averages from last one, five and fifteen minutes
- Memory Usage
  - Amount of free memory available to start new processes in KB.
  - High Swap Daemon Activity Percentage: When an Oracle Key Vault instance runs low on memory, the kernel memory swap daemon activates to swap the memory pages out, freeing up the memory. If the swap daemon exhibits high activity (indicated by significant CPU usage and ranking among the top three CPU-consuming processes) Oracle Key Vault considers this a high swap daemon activity which is an indication of high-memory pressure.
- Disk I/O
  - Number of disk read and write operations
- Network I/O
  - Amount of network data sent and received in KB
  - Rate of network data sent and received in KB/s
  - Number of incoming TCP connections

Required Authorization

System Administrator role or Monitor privilege

Syntax

```
okv metrics server get --start-time start_time --end-time end_time --include include --
interval interval --statistic statistic
```

JSON Input File Template

```
{
  "service" : {
    "category" : "metrics",
    "resource" : "server",
    "action" : "get",
    "options" : {
      "startTime" : "#YYYY-MM-DD hh:mm:ss",
      "endTime" : "#NOW|YYYY-MM-DD hh:mm:ss",
      "interval" : "#VALUE",
      "statistic" : "#MEAN|MAX|MIN|COUNT",
      "include" : [ "ALL" ],
      "limit" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--start-time`	Required	Specifies the start time for which results should be returned. It should be entered as timestamp in "YYYY-MM-DD HH:mm:ss" format. Example: Timestamp : YYYY-MM-DD HH:mm:ss

Parameter/Template Parameter	Required?	Description
`--end-time`	Required	The end time for which results should be returned. Example: Timestamp : YYYY-MM-DD HH:mm:ss or NOW.
`--interval`	Required	Specifies the time window used to convert given set of raw data points. Enter a value for this parameter using the ISO 8601 standard. For example, enter PT10M to specify an interval size of 10 Minutes. **Note** The timestamp of the aggregated data point corresponds to the start of the time window during which raw data points are assessed. For example, for a 10 minutes interval, the timestamp "09:00" corresponds to the 10 minutes time window from 09:00 to 09:10. Example: ISO 8601 Duration
`--statistic`	Required	Specifies the aggregation function applied to the given set of raw data points in an interval. Supported functions are MEAN, MIN, MAX, and COUNT. Example: MEAN, MIN, MAX, or COUNT.
`--include`	Required	Specifies the list of system resources that requires data. The supported resources are CPU, Memory, Disk, and network. Example: Specify "CPU" to include CPU, "CPU, MEMORY" to include CPU and Memory, "DISK" to include Disk, "NETWORK" to include Network, "ALL" to include all resources.
`--limit`	Optional	The maximum number of data should be returned.

JSON Example

1. Generate JSON input for the `okv metrics server get` command:

   ```
   okv metrics server get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated output to a file (for example, `metric_server_get.json`).

   ```
   {
     "service" : {
       "category" : "metrics",
       "resource" : "server",
       "action" : "get",
       "options" : {
   ```

```
      "startTime" : "2023-04-20 20:05:00",
      "endTime" : "NOW",
      "interval" : "PT10M",
      "statistic" : "MEAN",
      "include" : [ "ALL" ],
      "limit" : "3"
    }
  }
}
```

3. Run the `okv metrics server get` command using the generated JSON file:

```
okv metrics server get --from-json metric_server_get.json
```

Output similar to the following appears:


```
{
  "result" : "Success",
  "value" : {
    "availableResources" : [ {
      "sampleTime" : "2024-12-12 02:28:26",
      "cpuCores" : "2",
      "totalMemoryInKB" : "8112788"
    }, {
      "sampleTime" : "2024-12-12 03:53:08",
      "cpuCores" : "4",
      "totalMemoryInKB" : "32855068"
    } ],
    "metrics" : [ {
      "sampleTime" : "2024-12-12 02:28:26",
      "cpuUtilizationPercentage" : "99",
      "loadAverage1Minute" : "3.01",
      "loadAverage5Minutes" : "1.38",
      "loadAverage15Minutes" : "1.21",
      "freeMemoryInKB" : "1433548",
      "highSwapDaemonActivityPercentage" : "2",
      "diskReads" : "16763",
      "diskWrites" : "48480",
      "networkDataReceivedInKB" : "0",
      "networkDataSentInKB" : "0",
      "networkDataReceivedRateInKBps" : "0",
      "networkDataSentRateInKBps" : "0",
      "incomingTCPConnections" : "127"
    }, {
      "sampleTime" : "2024-12-12 03:53:08",
      "cpuUtilizationPercentage" : "73",
      "loadAverage1Minute" : "1.67",
      "loadAverage5Minutes" : "0.75",
      "loadAverage15Minutes" : "0.44",
      "freeMemoryInKB" : "26494624",
      "highSwapDaemonActivityPercentage" : "0",
      "diskReads" : "36189",
      "diskWrites" : "23666",
      "networkDataReceivedInKB" : "104245.51",
      "networkDataSentInKB" : "821488.99",
      "networkDataReceivedRateInKBps" : "17.05",
      "networkDataSentRateInKBps" : "84.13",
      "incomingTCPConnections" : "101"
    } ]
  }
}
```

okv primary-standby info get

The `okv primary-standby info get` command displays static information about the Oracle Key Vault primary-standby configuration.

`okv primary-standby info` retrieves the following information:

- The primary-standby status
- The primary server IP address
- The standby server IP address
- The fast-start failover threshold value

Required Authorization

System Administrator role or Monitor privilege

Syntax

```
okv primary-standby info get
```

JSON Input File Template

```
{
  "service" : {
    "category" : "primary-standby",
    "resource" : "info",
    "action" : "get"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv primary-standby info get` command:

   ```
   okv primary-standby info get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the output to a file (for example, `primary_standby_info.json`).
3. Run the `okv primary-standby info get` command using the generated JSON file:

   ```
   okv primary-standby info get --from-json primary_standby_info.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "fsfo" : "60",
       "primaryIPAddress" : "192.0.2.114",
       "primaryStandbyStatus" : "Primary",
       "standbyIPAddress" : "192.0.2.115"
   ```

```
    }
  }
```

okv primary-standby status get

The `okv primary-standby status get` command retrieves dynamic information about the Oracle Key Vault primary-standby configuration.

`okv primary-standby status get` get retrieves the following information:

- The switchover status
- The failover status
- Whether the primary is in read-only restricted mode

Required Authorization

System Administrator role or Monitor privilege

Syntax

```
okv primary-standby status get
```

JSON Input File Template

```
{
  "service" : {
    "category" : "primary-standby",
    "resource" : "status",
    "action" : "get"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv primary-standby status get` command:

   ```
   okv primary-standby status get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file (for example, `primary_standby_status.json`).
3. Run the `okv primary-standby status get` command using the generated JSON file:

   ```
   okv primary-standby status get --from-json primary_standby_status.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "failoverStatus" : "SYNCHRONIZED",
       "rormMode" : "No",
       "switchoverStatus" : "TO STANDBY"
     }
   }
   ```

okv server info get

The `okv server info get` command retrieves static information about an Oracle Key Vault server.

`okv server info get` displays the following:

- The current version of the Oracle Key Vault server
- The Oracle Key Vault server certification expiration date
- The deployment type of the Oracle Key Vault server, such as standalone, cluster, or primary-standby

Required Authorization

None. Information returned varies based on the user's authorization.

Syntax

```
okv server info get
```

JSON Input File Template

```
{
  "service" : {
    "category" : "server",
    "resource" : "info",
    "action" : "get"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv server info get` command:

   ```
   okv server info get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `server_info_get.json`). You do not need to edit the file because it has no parameters to modify.

3. Run the `okv server info get` command using the generated JSON file:

   ```
   okv server info get --from-json server_info_get.json
   ```

   Output similar to the following appears:

   ```
   As sysadmin:
   {
     "result" : "Success",
     "value" : {
       "caCertificateExpirationDate" : "2027-10-03 10:34:57",
       "cpuCores" : "16",
       "deploymentType" : "Cluster",
       "diskInGB" : "335",
   ```

```
    "fraInGB" : "160",
    "installDate" : "2024-10-03 10:31:33",
    "memoryInKB" : "7837396",
    "platformCertificatesExpirationDate" : "2025-10-03 10:33:11",
    "serverCertificateExpirationDate" : "2025-10-03 10:35:35",
    "serverTime" : "2024-10-09 16:13:03",
    "version" : "21.10.0.0.0"
  }
}

As user with no privileges:

{
  "result" : "Success",
  "value" : {
    "cpuCores" : "16",
    "deploymentType" : "Cluster",
    "diskInGB" : "335",
    "serverTime" : "2025-05-11 15:14:05",
    "version" : "21.10.0.0.0"
  }
}
```

All dates are shown in UTC.

okv server status get

The `okv server status get` command retrieves status information about an Oracle Key Vault server.

`okv server status get` displays the following:

- The amount of time (uptime) that the Oracle Key Vault has been running
- How much current free space is left on the Oracle Key Vault server
- The status of any backup jobs that have been started for Oracle Key Vault
- Number of alerts that have been raised concerning the Oracle Key Vault system
- Current CPU usage percentage and number of CPU cores installed on the Oracle Key Vault server
- Current disk space usage percentage and amount in GB and total disk space in GB on the Oracle Key Vault server
- Current Fast Recovery Area (FRA) space usage percentage, amount in GB, and total space in GB for FRA on the Oracle Key Vault server
- Current memory usage percentage, amount in KB, and total memory amount in KB installed on the Oracle Key Vault server
- Status of services running on the Oracle Key Vault server

Required Authorization

System Administrator role or Monitor privilege

Syntax

```
okv server status get
```

JSON Input File Template

```
{
  "service" : {
    "category" : "server",
    "resource" : "status",
    "action" : "get"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv server status get` command:

   ```
   okv server status get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `server_status.json`).

3. Run the `okv server status get` command using the generated JSON file:

   ```
   okv server status get --from-json server_status.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "alertsRaised" : "1",
       "backupStatus" : "25 day(s) since successful backup",
       "cpu" : {
         "usagePercentage" : "2",
         "cpuCores" : "16"
       },
       "disk" : {
         "usagePercentage" : "14",
         "usedInGB" : "42",
         "totalInGB" : "293"
       },
       "fra" : {
         "usagePercentage" : "20",
         "usedInGB" : "4",
         "totalInGB" : "20"
       },
       "freeSpacePercentage" : "86",
       "memory" : {
         "usagePercentage" : "98",
         "usedInKB" : "7655964",
         "totalInKB" : "7847768"
       },
       "services" : {
         "RESTfulService" : "Up",
         "emailService" : "Up",
         "KMIPService" : "Up",
         "storageDB" : "Up",
         "auditVaultAgentMonitor" : "Not enabled"
       },
   ```

```
    "uptime" : "11:59 HH:MM"
  }
}
```

> **Note**
>
> The count of `alertsRaised` may be different for users with the System Administrator role and the Monitor privilege.

8

Cluster Monitoring Commands

You can use the Cluster Management monitoring commands to check the Oracle Key Vault configuration, health, and deployment modes.

- okv cluster link disable
  The `okv cluster link disable` command disables a cluster node.
- okv cluster link enable
  The `okv cluster link enable` command enables the replication link between the current node and the given node in a cluster.
- okv cluster link monitor
  The `okv cluster link monitor` command monitors the replication link.
- okv cluster service monitor
  The `okv cluster service monitor` command monitors the status of a specific node or all nodes in a cluster.
- okv cluster service start
  The `okv cluster service start` command starts the cluster service of a node.
- okv cluster service stop
  The `okv cluster service stop` command stops the cluster service of a node.

okv cluster link disable

The `okv cluster link disable` command disables a cluster node.

Required Authorization

System Administrator role

`okv cluster link disable` command displays the following:

- The command result, whether it is Success or failure.

Syntax

```
okv cluster link disable
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "link",
    "action" : "enable",
    "options" : {
      "nodeName" : "#VALUE",
      "targetNode" : "#VALUE"
    }
```

```
  }
}
```

Parameters

Parameter	Required	Description
`--node-name`	Required	Name of the cluster node.
`--target-node`	Optional	Name of the node to initiate disabling of a node.

JSON Example

1. Generate JSON input for the `okv cluster link disable` command:

   ```
   okv cluster link disable --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `link_disable.json`.
3. Run the `okv cluster link disable` command using the generated JSON file. For example:

   ```
   okv cluster link disable --from-json link_disable.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv cluster link enable

The `okv cluster link enable` command enables the replication link between the current node and the given node in a cluster.

Required Authorization

System Administrator role

The `okv cluster link enable` command displays the command result - Success or Failure.

Syntax

```
okv cluster link enable --generate-json-input
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "link",
    "action" : "enable",
    "options" : {
      "nodeName" : "#VALUE",
```

```
      "targetNode" : "#VALUE"
    }
  }
}
```

Parameters

Parameter	Required	Description
`--node-name`	Required	Name of the cluster target node to enable.
`--target-node`	Optional	Name of the node to initiate disabling of a node

JSON Example

1. Generate JSON input for the `okv cluster link enable` command:

   ```
   okv cluster link enable
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `link_enable.json`.
3. Run the `okv cluster link enable` command using the generated JSON file. For example:

   ```
   okv cluster link enable --from-json link_enable.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv cluster link monitor

The `okv cluster link monitor` command monitors the replication link.

Required Authorization

System Administrator role

Syntax

```
okv cluster link monitor
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "link",
    "action" : "monitor",
    "options" : {
      "nodeName" : "#VALUE"
```

```
    }
  }
}
```

Parameters

Parameter	Required	Description
`--node-name`	Required	Name of the cluster node.

JSON Example

1. Generate JSON input for the `okv cluster link monitor` command:

   ```
   okv cluster link monitor --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `link_monitor.json`.
3. Run the `okv cluster link monitor` command using the generated JSON file. For example:

   ```
   okv cluster link monitor --from-json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "linkStatus":[ {
         "nodeID":"1",
         "nodeName":"okv080027ef30b1",
         "heartbeatLagInSeconds":"82.95",
         "linkState":"Up",
         "replicationLagInSeconds": "6"
       },
       {
         "nodeID":"2",
         "nodeName":"node2",
         "heartbeatLagInSeconds":"40.55",
         "linkState":"Up",
         "replicationLagInSeconds":"5"
       } ],
       "nodeID":"3",
       "nodeName":"node3"
     }
   }
   ```

okv cluster service monitor

The `okv cluster service monitor` command monitors the status of a specific node or all nodes in a cluster.

Required Authorization

System Administrator role

Syntax

```
okv cluster service monitor [--all-nodes] --node <node>
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "service",
    "action" : "monitor",
    "options" : {
      "nodeName" : "#VALUE",
      "allNodes" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter	Required	Description
`--node-name`	Required	Name of the cluster node.
`--all-nodes`	Optional	Shows the status of all nodes in the cluster (TRUE / FALSE).

JSON Example

1. Generate JSON input for the `okv cluster service monitor` command:

   ```
   okv cluster service monitor
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file, for example, `service_monitor.json`.

3. Run the `okv cluster service monitor` command using the generated JSON file. For example:

   ```
   okv cluster service monitor --from-json service_monitor.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "clusterServiceStatus":"Up",
       "nodeID":"2",
       "nodeName":"node2"
     }
   }
   ```

   Alternatively, if a value is set in the allnodes parameter, the following output will be displayed when all nodes are active:

```
{
  "result" : "Success",
  "value" : [ {
    "clusterServiceStatus" : "Up",
    "nodeID" : 1,
    "nodeName" : "node1"
  }, {
    "clusterServiceStatus" : "Up",
    "nodeID" : 2,
    "nodeName" : "node2"
  }, {
    "clusterServiceStatus" : "Up",
    "nodeID" : 3,
    "nodeName" : "node3"
  } ]
}
```

The following output will be displayed when node3 is disabled:

```
{
  "result" : "Success",
  "value" : [ {
    "clusterServiceStatus" : "Up",
    "nodeID" : 1,
    "nodeName" : "node1"
  }, {
    "clusterServiceStatus" : "Up",
    "nodeID" : 2,
    "nodeName" : "node2"
  }, {
    "clusterServiceStatus" : "Down",
    "nodeID" : 3,
    "nodeName" : "node3"
  } ]
}
```

okv cluster service start

The `okv cluster service start` command starts the cluster service of a node.

Required Authorization

System Administrator role

Syntax

```
okv cluster service start
```

JSON Input File Template

```
{
  "service" : {
    "category":"cluster",
    "resource":"service",
    "action":"start",
    "options":{
      "nodeName":"#VALUE"
    }
```

```
  }
}
```

Parameters

Parameter	Required	Description
`--node-name`	Required	Name of the cluster node.

JSON Example

1. Generate JSON input for the `okv cluster service start` command:

   ```
   okv cluster service start --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `service_start.json`.
3. Run the `okv cluster service start` command using the generated JSON file. For example:

   ```
   okv  cluster service start --from-json service_start.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv cluster service stop

The `okv cluster service stop` command stops the cluster service of a node.

Required Authorization

System Administrator role

Syntax

```
okv cluster service stop
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "service",
    "action" : "stop",
    "options" : {
      "nodeName" : "#VALUE"
    }
  }
}
```

Parameters

Parameter	Required	Description
`--node-name`	Required	Name of the cluster node

CLI Command

```
okv cluster service stop --node <argument>
```

JSON Example

1. Generate JSON input for the `okv cluster service stop` command:

   ```
   okv cluster service stop --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `service_stop.json`.
3. Run the `okv cluster service stop` command using the generated JSON file. For example:

   ```
   okv cluster service stop --from-json service_stop.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

9

Cluster Management Commands

You can use the Cluster Management commands to create a cluster, to manage a node in the cluster and to monitor the status of the node in the cluster.

- okv cluster node abort-pairing
  The `okv cluster node abort-pairing` command aborts the node pairing process in the cluster.
- okv cluster node add
  The `okv cluster node add` command adds a node to the cluster.
- okv cluster node cancel-disable
  The `okv cluster node cancel-disable` command cancels disabling of a cluster node.
- okv cluster node create
  The `okv cluster node create` command creates the first node of the cluster.
- okv cluster node delete
  The `okv cluster node delete` command deletes a cluster node.
- okv cluster node disable
  The `okv cluster node disable` command disables a cluster node.
- okv cluster node enable
  The `okv cluster node enable` command enables a cluster node.
- okv cluster node status
  The `okv cluster node status` command provides the information for the cluster node pairing process or the status of a cluster node.
- okv cluster node update
  The `okv cluster node update` command modifies the cluster subgroup of a cluster node.

okv cluster node abort-pairing

The `okv cluster node abort-pairing` command aborts the node pairing process in the cluster.

Required Authorization

System Administrator role

Syntax

```
okv cluster node abort-pairing --generate-json-input
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "node",
```

```
    "action" : "abort-pairing",
    "options" : {
      "candidateNodeIpAddress" : "#VALUE",
      "candidateNodeUser" : "#VALUE",
      "candidateNodePassword" : "#VALUE",
      "candidateNodePrivateIpAddress" :"#VALUE",
      "controllerNodePrivateIpAddress" :"#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required	Description
`--candidate-node-ip-address`	Optional	IP address of the candidate node.
`--candidate-node-password`	Optional	Password of the user from the candidate node. If `--candidate-node-password` is not provided, the process wait for the input.
`--candidate-node-private-ip-address`	Optional	Private IP address of the candidate node. **Note**: Provide the IP address for both the controller and candidate node if you are using the private IP address.
`--controller-node-private-ip-address`	Optional	Private IP address of the controller node. **Note**: Provide the IP address for both the controller and candidate node if you are using the private IP address.
`--candidate-node-user`	Optional	Name of a user from the candidate node.

> **Note**
>
> If you do not provide information in the `--candidate-node-ip-address`, the command gets aborted in the controller node. If you want to abort the `--candidate-node-ip-address`, you have to provide information for all the three parameters.

CLI Command

```
okv cluster node abort-pairing --options <argument>
```

Usage Notes

Once you submit the add or create command to view the status, perform the following steps:

1. You can get `requestID`. Using this `requestID`, you can check the request status, using the command:

   ```
   okv cluster node status --pairing-request-id
   ```

2. Following this, you can check the node current status using the command:

   ```
   okv cluster node status
   ```

JSON Example

1. Generate JSON input for the `okv cluster node abort-pairing` command.

   ```
   okv cluster node abort-pairing
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `abort_pairing.json`.
3. Run the `okv cluster node abort-pairing` command using the generated JSON file. For example:

   ```
   okv cluster node abort-pairing --from-json abort_pairing.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "requestId" : "78223"
     }
   }
   ```

> **Note**
>
> See okv cluster node status for requestId output.

Related Topics

- okv cluster node status
  The `okv cluster node status` command provides the information for the cluster node pairing process or the status of a cluster node.

okv cluster node add

The `okv cluster node add` command adds a node to the cluster.

Syntax

```
okv cluster node add
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "node",
    "action" : "add",
    "options" : {
      "recoveryPassphrase" : "#VALUE",
      "candidateNodeIpAddress" : "#VALUE",
      "mode" : "#VALUE",
      "nodeId" : "#VALUE",
      "nodeName" : "#VALUE",
      "clusterSubgroup" : "#VALUE",
      "hsmCredential" : "#VALUE",
      "candidateNodeUser" : "#VALUE",
      "candidateNodePassword" : "#VALUE",
      "candidateNodePrivateIpAddress" : "#VALUE",
      "controllerNodePrivateIpAddress" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--candidate-node-ip-address`	Required	IP address of the candidate node.
`--candidate-node-password`	Required	Password of the user from the candidate node. **Note** The user is not required to provide the password in command line or JSON. The user will be prompted for password during runtime.
`--candidate-node-user`	Required	Name of a user from the candidate node.
`--candidate-node-private-ip-address`	Optional	Private IP address of the candidate node. **Note**: Provide the private IP address for both the controller and candidate node if you are using the private IP address.
`--controller-node-private-ip-address`	Optional	Private IP address of the controller node. **Note**: Provide the private IP address for both the controller and candidate node if you are using the private IP address.
`--cluster-subgroup`	Required	Name of the cluster subgroup.
`--mode`	Required	Pairing mode - READ-ONLY or READ-WRITE.
`--node-id`	Required	ID of the cluster node. The id can be between 1 to 16 and the ID should be unique and not the duplicated ID with other node.

Parameter/Template Parameter	Required?	Description
`--node-name`	Required	Name of the cluster node.
`--recovery-passphrase`	Required	Recovery pass phrase of the cluster.
		Note The user is not required to provide the password in command line or JSON. The user will be prompted for password during runtime.
`--hsm-credential`	Optional	HSM credential.

CLI Command

```
okv cluster node add --options <argument>
```

JSON Example

1. Generate JSON input for the `okv cluster node add` command:

   ```
   okv cluster node add --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `node-add.json`.
3. Run the `okv cluster node add` command using the generated JSON file. For example:

   ```
   okv cluster node add --from-json-node-add.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "requestId" : "78219"
     }
   }
   ```

Note

See, section okv cluster node status for requestId output.

Related Topics

- okv cluster node delete
  The `okv cluster node delete` command deletes a cluster node.
- okv cluster node status
  The `okv cluster node status` command provides the information for the cluster node pairing process or the status of a cluster node.

okv cluster node cancel-disable

The `okv cluster node cancel-disable` command cancels disabling of a cluster node.

Required Authorization

System Administrator role

`okv cluster node cancel-disable` command displays the following:

- The command result, whether it is Success or failure.

Syntax

```
okv cluster node cancel-disable
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "node",
    "action" : "cancel-disable",
    "options" : {
      "nodeName" : "#VALUE"
    }
  }
}
```

Parameters

Parameter	Required?	Description
`--node-name`	Required	Name of the cluster node.

JSON Example

1. Generate JSON input for the `okv cluster node cancel-disable` command:

   ```
   okv cluster node cancel-disable --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `cancel_disable.json`.

3. Run the `okv cluster node cancel-disable` command using the generated JSON file. For example:

```
okv cluster node cancel-disable --from-json cancel_disable.json
```

Output similar to the following appears:

```
{
  "result" : "Success"
}
```

okv cluster node create

The `okv cluster node create` command creates the first node of the cluster.

Purpose

The `okv cluster node create` command is the first node under cluster. Following information is required:

- cluster name
- cluster node name
- subgroup name in a cluster

Syntax

```
okv cluster node create --generate-json-input
```

JSON Input File Template

```
{
  "service" : {
   "category" : "cluster",
   "resource" : "node",
   "action" : "create",
   "options" : {
     "nodeName" : "#VALUE",
     "clusterName" : "#VALUE",
     "clusterSubgroup" : "#VALUE"
     }
   }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--cluster-name`	Required	Name of the cluster.
`--cluster-subgroup`	Required	Name of the cluster subgroup.
`--node-name`	Required	Name of the cluster node.

CLI Command

```
okv cluster node create --cluster-name --cluster-subgroup --node-name
```

CLI Example

```
okv cluster node create --cluster-name "NorthAmerica" --cluster-subgroup
"NewYork" --node-name "node1"
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "requestId" : "1630"
  }
}
```

Usage Notes

Once you submit the add or create command to view the status, perform the following steps:

1. You can get `requestID`. Using this `requestID`, you can check the request status, using the command:

   ```
   okv cluster node status --pairing-request-id
   ```

2. Following this, you can check the current node status using the command:

   ```
   okv cluster node status
   ```

JSON Example

1. Generate JSON input for the `okv cluster node create` command:

   ```
   okv cluster node create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file, for example, `node-create.json`.

3. Run the `okv cluster node create` command using the generated JSON file. For example, `okv cluster node create --from-json node-create.json`.
   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "requestId" : "1926"
     }
   }
   ```

Related Topics

- okv cluster node status
  The `okv cluster node status` command provides the information for the cluster node pairing process or the status of a cluster node.

okv cluster node delete

The `okv cluster node delete` command deletes a cluster node.

Required Authorization

System Administrator role

`okv cluster node delete` command displays the following:

- The command result, whether it is Success or failure.

Syntax

```
okv cluster node delete
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "node",
    "action" : "delete",
    "options" : {
      "nodeName" : "#VALUE",
      "force" : "#TRUE|FALSE",
      "originNodeName" : "#VALUE"
    }
  }
}
```

Parameter	Required	Description
`--node-name`	Required	Name of the cluster node.
`--origin-node-name`	Required	Name of an origin node to initiate deletion of a node.
`--force`	Optional	Use force option to delete a node - TRUE or FALSE.

CLI Command

```
okv cluster node delete --options <argument>
```

JSON Example

1. Generate JSON input for the `okv cluster node delete` command:

   ```
   okv cluster node delete --generate-json-input
   ```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file, for example, `node_delete.json`.
3. Run the `okv cluster node delete` command using the generated JSON file. For example:

```
okv cluster node delete --from-json node_delete.json
```

Output similar to the following appears:

```
{
  "result" : "Success"
}
```

> **Note**
>
> See, section okv cluster node status for requestId output.

okv cluster node disable

The `okv cluster node disable` command disables a cluster node.

Required Authorization

System Administrator role

Syntax

```
okv cluster node disable
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "node",
    "action" : "disable",
    "options" : {
      "nodeName" : "#VALUE",
      "originNodeName" : "#VALUE"
    }
  }
}
```

Parameters

Parameter	Required?	Description
`--node-name`	Required	Name of the cluster node.
`--origin-node-name`	Optional	Name of an origin node to initiate disabling of a node.

JSON Example

1. Generate JSON input for the `okv cluster node disable` command:

   ```
   okv cluster node disable --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `node_disable.json`.
3. Run the `okv cluster node disable` command using the generated JSON file. For example:

   ```
   okv cluster node disable --from-json node_disable.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" :"Success"
   }
   ```

okv cluster node enable

The `okv cluster node enable` command enables a cluster node.

Required Authorization

System Administrator role

`okv cluster node enable` command displays the following:

- The command result, whether it is Success or failure.

Syntax

```
okv cluster node enable
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "node",
    "action" : "enable",
    "options" : {
      "nodeName" : "#VALUE"
    }
  }
}
```

Parameters

Parameter	Required	Description
`--node-name`	Required	Name of the cluster node.

JSON Example

1. Generate JSON input for the `okv cluster node enable` command:

   ```
   okv cluster node enable --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file, for example, `node_enable.json`.
3. Run the `okv cluster node enable` command using the generated JSON file. For example:

   ```
   okv cluster node enable --from-json node_enable.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv cluster node status

The `okv cluster node status` command provides the information for the cluster node pairing process or the status of a cluster node.

Required Authorization

System Administrator role

Syntax

```
okv cluster node status
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "node",
    "action" : "status",
    "options" : {
      "node-name" : "#VALUE",
      "pairingRequestId" : "#VALUE",
      "pairingSteps" : "#TRUE|FALSE",
      "candidateNodeIpAddress" : "#VALUE",
      "candidateNodeUser" : "#VALUE",
      "candidateNodePassword" : "#VALUE",
      "candidateNodePrivateIpAddress" :"#VALUE",
      "controllerNodePrivateIpAddress" :"#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required	Description
`--candidate-node-ip-address`	Optional	IP address of the candidate node.
`--candidate-node-password`	Optional	Password of the user from the candidate node.
`--candidate-node-user`	Optional	Name of a user from the candidate node.
`--candidate-node-private-ip-address`	Optional	Private IP address of the candidate node. **Note**: Provide the IP address for both the controller and candidate node if you are using the private IP address.
`--controller-node-private-ip-address`	Optional	Private IP address of the controller node. **Note**: Provide the IP address for both the controller and candidate node if you are using the private IP address.
`--node-name`	Optional	Name of the cluster node.
`--pairing-request-id`	Optional	Request ID for a long running command.
`--pairing-steps`	Optional	Display detailed pairing steps - TRUE or FALSE.

CLI Command

```
1. $ okv cluster node status
2. $ okv cluster node status --node-name
3. $ okv cluster node status --pairing-request-id
4. $ okv cluster node status --pairing-steps TRUE
5. $ okv cluster node status --pairing-steps --candidate-node-ip-address --
candidate-node-user --candidate-node-password
```

> **Note**
>
> You need to follow the multiple node in the provided sequence:
>
> 1. You can use `okv cluster node status` command for candidate node or controller node that belongs to manage cluster commands. The command will check the status of controller node.
> 2. You can use `okv cluster node status --pairing-request-id` for commands that returns `requestId`. The command will check the status of the given node.
> 3. You can use `okv cluster node status --pairing-steps` to check pairing step in controller node for an "add" command. It will check the status of the job by a request ID. The request ID is available from "create", "add", and "abort-paring) command. The command will check the status of pairing steps in controller node.
> 4. You can use `okv cluster node status --pairing-steps --candidate-node-ip-address --candidate-node-user --candidate-node-password` to check pairing steps in candidate node for an `add` command. The command will check the status of pairing steps in controller node.

Usage Notes

Once you submit the add or create command to view the status, perform the following steps:

1. You can get `requestID`. Using this `requestID`, you can check the request status, using the command:

   ```
   okv cluster node status --pairing-request-id
   ```

2. Following this, you can check the node current status using the command:

   ```
   okv cluster node status
   ```

JSON Example

1. Generate JSON input for the `okv cluster node status` command:

   ```
   okv cluster node status --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file, for example, `node_status.json`.
3. Run the `okv cluster node status` command using the generated JSON file. For example:

   ```
   okv cluster node status --from-json node_status.json
   ```

   Output similar to the following appears:

   ```
   okv cluster node status:

   {
     "result" : "Success",
     "value" : {
       "status" : "CONFIGURED"
   ```

```
  }
}

okv cluster node status --node-name node1
{
  "result" : "Success",
  "value" : {
    "status" : "ENABLING"
  }
}

okv cluster node status --pairing-request-id 18374:
{
  "result" : "Failure",
  "message" : "Server is already a node or is already configured as node."
}
{
  "result" : "Success"
}

okv cluster node status  --pairing-steps TRUE
{
  "result" : "Success",
  "value" : {
    "stages" : [ {
      "step1" : "Open transport channel with the candidate node",
      "status" : "COMPLETED"
    }, {
      "step2" : "Verify the candidate node details",
      "status" : "COMPLETED"
    }, {
      "step3" : "Enable data replication to the candidate node",
      "status" : "COMPLETED"
    }, {
      "step4" : "Generate the controller node details",
      "status" : "COMPLETED"
    }, {
      "step5" : "Generate backup of the controller node for cloning",
      "status" : "COMPLETED"
    }, {
      "step6" : "Send clone bundle to the candidate node",
      "status" : "COMPLETED"
    }, {
      "step8" : "Enable data replication to other cluster nodes",
      "status" : ""
    }, {
      "step9" : "The candidate node successfully joins the cluster",
      "status" : ""
    } ]
  }
}
okv cluster node status --pairing-steps TRUE --candidate-node-ip-
address100.70.126.53 --candidate-node-user okvadmin --candidate-node-password
Welcome_1
{
  "result" : "Success",
  "value" : {
    "stages" : [ {
      "step1" : "Generate the candidate node details",
      "status" : "COMPLETED"
    }, {
      "step2" : "Open transport channel with the controller node",
```

```
        "status" : "COMPLETED"
      }, {
        "step3" : "Send node details to the controller node",
        "status" : "COMPLETED"
      }, {
        "step4" : "Receive clone bundle from the controller node",
        "status" : "COMPLETED"
      }, {
        "step5" : "Restore backup on the candidate node",
        "status" : "COMPLETED"
      }, {
        "step6" : "Update credentials of the candidate node",
        "status" : "COMPLETED"
      }, {
        "step7" : "Tune the database on the candidate node",
        "status" : "COMPLETED"
      }, {
        "step8" : "Setup network configuration on the candidate node",
        "status" : "COMPLETED"
      }]
    }
  }
```

okv cluster node update

The `okv cluster node update` command modifies the cluster subgroup of a cluster node.

Required Authorization

System Administrator role

Syntax

```
okv cluster node update --generate-json-input
```

JSON Input File Template

```
{
  "service" : {
    "category" : "cluster",
    "resource" : "node",
    "action" : "update",
    "options" : {
      "nodeName" : "#VALUE",
      "clusterSubgroup" : "#VALUE"
    }
  }
}
```

Parameters

Parameter	Required	Description
`--cluster-subgroup`	Required	Name of the cluster subgroup.
`--node-name`	Required	Name of the cluster node.

JSON Example

1. Generate JSON input for the `okv cluster node update` command:

   ```
   okv cluster node update
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file, for example, `node_update.json`.

3. Run the `okv cluster node delete` command using the generated JSON file. For example:

   ```
   okv cluster node update --from-json node_update.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

10

Backup, Schedule, and Restore Commands

You can use the backup, schedule, and restore commands to automate Oracle Key Vault appliance backups.

- okv backup destination create
  The `okv backup destination create` command creates a remote backup destination for the Oracle Key Vault server.
- okv backup destination delete
  The `okv backup destination delete` command deletes a remote backup destination.
- okv backup destination delete-backup
  The `okv backup destination delete-backup` command enables you to manually delete a local Oracle Key Vault backup.
- okv backup destination get
  The `okv backup destination get` command gets information about a specific backup destination.
- okv backup destination get-public-key
  The `okv backup destination get-public-key` command retrieves the SSH public key of the Oracle Key Vault internal user used for performing backups.
- okv backup destination list
  The `okv backup destination list` command displays a list of the current Oracle Key Vault server backup destinations.
- okv backup destination list-backups
  The `okv backup destination list-backups` command lists the backups that are available for restore operations on a destination.
- okv backup destination reset-host-key
  The `okv backup destination reset-host-key` command resets a destination host's public key in the `known_hosts` file for the `oracle` user.
- okv backup destination resume-policy
  The `okv backup destination resume-policy` command resumes the operation of the backup destination policy that is associated with a specified backup destination.
- okv backup destination suspend-policy
  The `okv backup destination suspend-policy` command suspends the operation of the backup destination policy that is associated with a specified backup destination.
- okv backup destination update
  The `okv backup destination update` command updates the settings of a remote backup destination.
- okv backup destination-policy create
  The `okv backup destination-policy create` command creates a backup destination policy.
- okv backup destination-policy delete
  The `okv backup destination-policy delete` command deletes a backup destination policy.

- okv backup destination-policy get
  The `okv backup destination-policy get` command retrieves detailed information about a backup destination policy.
- okv backup destination-policy list
  The `okv backup destination-policy list` command lists existing backup destination policies and their settings.
- okv backup destination-policy list-purged-backups
  The `okv backup destination-policy list-purged-backups` command lists the backups purged by a backup destination policy.
- okv backup destination-policy update
  The `okv backup destination-policy update` command updates a backup destination policy.
- okv backup history list
  The `okv backup history list` command lists the details of a backup history, such as runtime errors, whether the backup completed, and start and end times.
- okv backup restore start
  The `okv backup restore start` command starts the restore process of an Oracle Key Vault backup.
- okv backup restore status
  The `okv backup restore status` command checks the status of the Oracle Key Vault backup restore operation.
- okv backup schedule create
  The `okv backup schedule create` command creates a backup schedule job.
- okv backup schedule delete
  The `okv backup schedule delete` command deletes scheduled backup job.
- okv backup schedule get
  The `okv backup schedule get` command retrieves detailed information about a scheduled Oracle Key Vault server backup.
- okv backup schedule list
  The `okv backup schedule list` command displays a listing of the currently scheduled Oracle Key Vault server backups.
- okv backup schedule pause
  The `okv backup schedule pause` command pauses a scheduled Oracle Key Vault server backup.
- okv backup schedule resume
  The `okv backup schedule resume` command resumes a paused Oracle Key Vault backup job.
- okv backup schedule run-now
  The `okv backup schedule run-now` command runs the backup schedule job immediately.
- okv backup schedule update
  The `okv backup schedule update` command updates a currently scheduled backup.

okv backup destination create

The `okv backup destination create` command creates a remote backup destination for the Oracle Key Vault server.

Required Authorization

System Administrator role

Syntax

```
okv backup destination create
--name destination_name
--transfer-method scp|sftp
--host-name host_name
--port port
--path destination_path
--username user_name
--authentication-method password|key-based
--destination-policy policy_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "create",
    "options" : {
      "name" : "#VALUE",
      "transferMethod" : "#scp|sftp",
      "hostName" : "#VALUE",
      "port" : "#VALUE",
      "path" : "#VALUE",
      "userName" : "#VALUE",
      "authenticationMethod" : "#password|key-based",
      "destinationPolicy" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup destination. For a listing of existing names, run the `okv backup destination list` command.
`--transfer-method` / `transferMethod`	Optional	Enter one of the following values: • `scp` • `sftp` (Default value)
`--host-name` / `hostName`	Required	Host name or IP address of the remote server for the backup. If you enter the host name, then ensure that DNS is configured to translate the host name to its corresponding IP address. Do not include spaces, single quotation marks, or double quotation marks in a host name that is in a remote backup destination.

Parameter/Template Parameter	Required?	Description
`--port` / `port`	Optional	Port number of the destination backup computer. The default is 22.
`--path` / `path`	Required	Path to an existing directory on the external server where the backup file will be copied. You cannot modify this directory location after the backup destination is created. This path must not be the destination for backups from another Oracle Key Vault server. Do not include spaces, single quotation marks, or double quotation marks destination path that is in a remote backup destination.
`--username` / `userName`	Required	User name of the user account on the remote server. Ensure that this user has the write permissions on the directory specified in the path parameter for the `scp` connection. Do not include spaces, single quotation marks, or double quotation marks in a user name that is in a remote backup destination.
`--authentication-method` / `authenticationMethod`	Optional	Enter one of the following values: • `password`: The password of the user account specified in the `--username` / `userName` parameter. • `key-based` (Default value): Use `okv backup destination get-public-key` to obtain the public key of the Oracle Key Vault internal user. Copy the public key from the command output and paste it in the appropriate configuration file, such as `authorized_keys`, on the destination server. Check that the permissions of the configuration file are set to allow access only to the backup account owner and no other group or user. Be aware that certain events may trigger a change of the public key, which means that Oracle Key Vault cannot use the backup destination until the new public key is re-copied from Oracle Key Vault to the appropriate configuration file. These events include but are not limited to certificate rotation, changing the IP address, and conversion to a cluster node.
`--destination-policy` / `destinationPolicy`	Optional	Specifies the name of the backup destination policy. To find existing backup destination policies, run the `okv backup destination-policy list` command.

JSON Example

1. Generate JSON input for the `okv backup destination create` command:

   ```
   okv backup destination create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_srvr_dest_create.json`) and then edit it to create the backup server destination:

   ```
   {
     "service" : {
   ```

```
    "category" : "backup",
    "resource" : "destination",
    "action" : "create",
    "options" : {
      "name" : "prod_okv_backup_dest",
      "transferMethod" : "scp",
      "hostName" : "192.0.2.34",
      "port" : "22",
      "path" : "/opt/okv/backups",
      "userName" : "psmith",
      "authenticationMethod" : "password",
      "destinationPolicy" : "global_dest_pol"
    }
  }
}
```

3. Run the `okv backup destination create` command using the generated JSON file:

   ```
   okv backup destination create --from-json bkup_srvr_dest_create.json
   ```

   If you specified `password` for the user authentication method, then you will be prompted for the password. After entering the correct password, output similar to the following appears:

   ```
   Destination User Password: password
   {
     "result" : "Success"
   }
   ```

okv backup destination delete

The `okv backup destination delete` command deletes a remote backup destination.

Required Authorization

System Administrator role

Syntax

```
okv backup destination delete --name destination_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "delete",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template	Required?	Description
`--name` / `name`	Required	Name of the remote backup destination. To find existing backup names, run the `okv backup destination list` command.

JSON Example

1. Generate JSON input for the `okv backup destination delete` command:

   ```
   okv backup destination delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_srvr_dest_del.json`) and then edit it to delete the backup server destination:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "delete",
       "options" : {
         "name" : "prod_okv_backup_dest"
       }
     }
   }
   ```

3. Run the `okv backup destination delete` command using the generated JSON file:

   ```
   okv backup destination delete --from-json bkup_srvr_dest_del.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup destination delete-backup

The `okv backup destination delete-backup` command enables you to manually delete a local Oracle Key Vault backup.

Required Authorization

System Administrator role

Syntax

```
okv backup destination delete-backup --destination backup_destination --backup-file
backup_file_name
```

JSON Input File Template

```
{
  "service" : {
     "category" : "backup",
     "resource" : "destination",
     "action" : "delete-backup",
     "options" : {
       "backupFile" : "#VALUE",
       "destination" : "#VALUE"
     }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--destination` / `destination`	Required	Specifies the name of the backup destination. Enter `LOCAL`. The `okv backup destination delete-backup` command deletes an Oracle Key Vault backup by its backup file name. This is only supported for `LOCAL` destination.
`--backup-file` / `backupFile`	Required	Specifies the name of the backup file. To find existing backup file names, run the `okv backup destination list-backups` command.

JSON Example

1. Generate JSON input for the `okv backup destination delete-backup` command:

   ```
   okv backup destination delete-backup --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_local_del.json`) and then edit it to specify the backup file name and backup destination name (that is, `local`):

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "delete-backup",
       "options" : {
         "backupFile" : "okvbackup_onetime_onetime_20210118175804",
         "destination" : "local"
       }
     }
   }
   ```

3. Run the `okv backup destination delete-backup` command using the generated JSON file:

   ```
   okv backup destination delete-backup --from-json backup_local_del.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup destination get

The `okv backup destination get` command gets information about a specific backup destination.

Required Authorization

System Administrator role

Syntax

```
okv backup destination get --name backup_destination_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "get",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup destinination. For a listing of existing names, run the `okv backup destination list` command.

JSON Example

1. Generate JSON input for the `okv backup destination get` command:

   ```
   okv backup destination get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_srvr_dest_get.json`) and then edit it to retrieve the backup server destination:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "get",
       "options" : {
         "name" : "prod_okv_backup_dest"
       }
     }
   }
   ```

3. Run the `okv backup destination get` command using the generated JSON file:

   ```
   okv backup destination get --from-json bkup_srvr_dest_get.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "authenticationMethod" : "Password",
       "hostName" : "192.0.2.34",
       "name" : "PROD_OKV_BACKUP_DEST",
       "path" : "/opt/okv/backups",
   ```

```
    "policyName" : "GLOBAL_DEST_POL",
    "policyState" : "Active",
    "port" : "22",
    "transferMethod" : "scp",
    "userName" : "psmith"
  }
}
```

okv backup destination get-public-key

The `okv backup destination get-public-key` command retrieves the SSH public key of the Oracle Key Vault internal user used for performing backups.

Copy the public key from the output of this command and paste it in the appropriate configuration file, such as `authorized_keys` of the backup destination user account, on the backup destination server. Check that the permissions of the configuration file are set to allow access only to the backup user account and to no other group or user. Be aware that certain events may trigger a change of the public key, which means that Oracle Key Vault cannot use the backup destination until the new public key is re-copied from Oracle Key Vault to the appropriate configuration file. These events include but are not limited to certificate rotation, changing the IP address, and conversion to a cluster node.

Required Authorization

System Administrator role

Syntax

```
okv backup destination get-public-key
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "get-public-key"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv backup destination get-public-key` command:

   ```
   okv backup destination get-public-key --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_dest_get_public_key.json`).

3. Run the `okv backup destination get-public-key` command using the generated JSON file:

   ```
   okv backup destination get-public-key --from-json bkup_dest_get_public_key.json
   ```

   Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "object" : "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDRLparTgrf5F2IExZ1IMZecobMw2ptj5WWlr6l2ww9GHZ5YgMiTNBCi
Ajr68KgLgJ9eRkOSSz7tsnYzcw8th45abB344LzMBLREqtqbV4U0PYHMmt1ovhd+djhsYnJbXptfiSAfe2f+1
/XPlIYcZNo3m5imffgaIsrn9WlIYxOnP7rrZW3mQkPRLADElTAMWl7zDj71mZrenNBInTd70CBX/
L7C4NABikPu1E7TxpASQRW9y/n5zdGR4TVww06nAEseCfwfzV1ToNK7CFwFWv/OdIARVVSqwCkCDrwP/
pNYr7WjzXR939xBfuXaWNZpoDkN1Yxb5sk1NEYRT+cs/SD\n"
  }
}
```

okv backup destination list

The `okv backup destination list` command displays a list of the current Oracle Key Vault server backup destinations.

Required Authorization

System Administrator role

Syntax

```
okv backup destination list
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "list"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv backup destination list` command:

   ```
   okv backup destination list --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_srvr_dest_list.json`).

3. Run the `okv backup destination list` command using the generated JSON file:

   ```
   okv backup destination list --from-json bkup_srvr_dest_list.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "destinations" : [
       {
         "authenticationMethod" : " ",
         "hostName" : "localhost",
   ```

```
      "name" : "LOCAL",
      "path" : "-",
      "transferMethod" : "-",
      "userName" : "-"
    },
    {
      "authenticationMethod" : "Password",
      "hostName" : "192.0.2.34",
      "name" : "PROD_OKV_BACKUP_DEST",
      "path" : "/opt/okv/backups",
      "port" : "22",
      "transferMethod" : "scp",
      "userName" : "psmith"
    }
  ]
}
```

okv backup destination list-backups

The `okv backup destination list-backups` command lists the backups that are available for restore operations on a destination.

Required Authorization

System Administrator role

Syntax

`okv backup destination list-backups --name` *`destination_name`*

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "list-backups",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup destination. To find existing backup destination names, run the `okv backup destination list` command.

JSON Example

1. Generate JSON input for the `okv backup destination list-backups` command:

   ```
   okv backup destination list-backups --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `dest_list_backups.json`) and then edit it to list the backup server destinations:

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "list-backups",
    "options" : {
      "name" : "prod_okv_backup_dest"
    }
  }
}
```

3. Run the `okv backup destination list-backups` command using the generated JSON file:

```
okv backup destination list-backups --from-json dest_list_backups.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "backups" : [
    {
      "file" : "okvbackup_periodic_incr_20211030162206",
      "time" : "2021-10-30 16:22:06",
      "type" : "Periodic"
    }, {
      "file" : "okvbackup_periodic_full_20211029162206",
      "time" : "2021-10-29 16:22:06",
      "type" : "Periodic"
    }, {
      "file" : "okvbackup_onetime_onetime_20211028012206",
      "time" : "2021-10-28 01:22:06",
      "type" : "One-Time"
    }
  ]
}
```

okv backup destination reset-host-key

The `okv backup destination reset-host-key` command resets a destination host's public key in the `known_hosts` file for the `oracle` user.

Required Authorization

System Administrator role

Syntax

```
okv backup destination reset-host-key --name destination_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "reset-host-key",
    "options" : {
      "name" : "#VALUE"
```

```
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup destination. To find existing backup destination names, run the `okv backup destination list` command.

JSON Example

1. Generate JSON input for the `okv backup destination reset-host-key` command:

   ```
   okv backup destination reset-host-key --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `reset_host_key.json`) and then edit it to reset the host key for the destination server:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "reset-host-key",
       "options" : {
         "name" : "prod_okv_backup_dest"
       }
     }
   }
   ```

3. Run the `okv backup destination reset-host-key` command using the generated JSON file:

   ```
   okv backup destination reset-host-key --from-json reset_host_key.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup destination resume-policy

The `okv backup destination resume-policy` command resumes the operation of the backup destination policy that is associated with a specified backup destination.

Required Authorization

System Administrator role

Syntax

```
okv backup destination resume-policy --destination destination_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "resume-policy",
    "options" : {
      "destination" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--destination` / `destination`	Required	Specifies the name of the backup destination. For a listing of existing destination names, run the `okv backup destination list` command.

JSON Example

1. Generate JSON input for the `okv backup destination resume-policy` command:

   ```
   okv backup destination resume-policy --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_dest_pol_resume.json`) and then edit it to specify the backup destination policy to resume:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "resume-policy",
       "options" : {
         "destination" : "prod_okv_backup_dest"
       }
     }
   }
   ```

3. Run the `okv backup destination resume-policy` command using the generated JSON file:

   ```
   okv backup destination resume-policy --from-json backup_dest_pol_resume.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup destination suspend-policy

The `okv backup destination suspend-policy` command suspends the operation of the backup destination policy that is associated with a specified backup destination.

Required Authorization

System Administrator role

Syntax

```
okv backup destination suspend-policy --destination destination_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "suspend-policy",
    "options" : {
      "destination" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--destination` / `destination`	Required	Name of the backup destination. For a listing of existing backup destination names, run the `okv backup destination list` command.

JSON Example

1. Generate JSON input for the `okv backup destination suspend-policy` command:

   ```
   okv backup destination suspend-policy --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_dest_pol_suspend.json`) and then edit it to specify the backup destination policy to suspend:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "suspend-policy",
       "options" : {
         "destination" : "prod_okv_backup_dest"
       }
     }
   }
   ```

3. Run the `okv backup destination suspend-policy` command using the generated JSON file:

```
okv backup destination suspend-policy --from-json backup_dest_pol_suspend.json
```

Output similar to the following appears:

```
{
  "result" : "Success"
}
```

okv backup destination update

The `okv backup destination update` command updates the settings of a remote backup destination.

Required Authorization

System Administrator role

Syntax

```
okv backup destination update
--name destination_name
--port port
--user-name user_name
--authentication-method authentication_method
--destination-policy destination_policy_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "update",
    "options" : {
      "name" : "#VALUE",
      "port" : "#VALUE",
      "userName" : "#VALUE",
      "authenticationMethod" : "#password|key-based",
      "destinationPolicy" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup destination. For a listing of existing backup destination names, run the `okv backup destination list` command.
`--port` / `port`	Optional	Port number of the destination backup computer. The default is 22.
`--user-name` / `userName`	Optional	User name of the user account on the remote server. Ensure that this user has the write permissions on the directory specified in the path parameter for the `scp` connection. Do not include spaces, single quotation marks, or double quotation marks in a user name that is in a remote backup destination.

Parameter/Template Parameter	Required?	Description
`--authentication-method` / `authenticationMethod`	Optional	Enter one of the following values: • `password`: The password of the user account specified in the `--user-name` / `userName`parameter. The default value is: • `key-based`: Use `okv backup destination get-public-key` to obtain the public key of the Oracle Key Vault internal user. Copy the public key from the command output and paste it in the appropriate configuration file, such as `authorized_keys`, on the destination server. Check that the permissions of the configuration file are set to allow access only to the backup account owner and no other group or user. Be aware that certain events may trigger a change of the public key, which means that Oracle Key Vault cannot use the backup destination until the new public key is re-copied from Oracle Key Vault to the appropriate configuration file. These events include but are not limited to certificate rotation, changing the IP address, and conversion to a cluster node.
`--destination-policy` / `destinationPolicy`	Optional	Specifies the backup destination policy. To find existing backup destination policies, run the `okv backup destination-policy list` command.

JSON Example

1. Generate JSON input for the `okv backup destination update` command:

   ```
   okv backup destination update --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `update_bkup_srvr_dest.json`) and then edit it to update the backup server destination:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "update",
       "options" : {
         "name" : "prod_okv_backup_dest",
         "port" : "22",
         "userName" : "psmith",
         "authenticationMethod" : "password",
         "destinationPolicy" : "rac_dest_pol"
       }
     }
   }
   ```

3. Run the `okv backup destination update` command using the generated JSON file:

   ```
   okv backup destination update --from-json update_bkup_srvr_dest.json
   ```

If you specified `password` for the user authentication method, then you will be prompted for the password. After entering the correct password, output similar to the following appears:

```
Destination User Password: password
{
  "result" : "Success"
}
```

okv backup destination-policy create

The `okv backup destination-policy create` command creates a backup destination policy.

Required Authorization

System Administrator role

Syntax

```
okv backup destination-policy create --destination-policy policy_name --recent-backups-
to-preserve number_of_recent_backups_to_preserve --purge-backup-after
number_of_days_after_which_backup_purged
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination-policy",
    "action" : "create",
    "options" : {
      "destinationPolicy" : "#VALUE",
      "recentBackupsToPreserve" : "#VALUE",
      "purgeBackupAfter" :  "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--destination-policy` / `destinationPolicy`	Required	Specifies the name of the backup destination policy to create. To find existing backup destination policies, run the `okv backup destination-policy list` command.
`--recent-backups-to-preserve` / `recentBackupsToPreserve`	Optional	The number of the most recent backups to preserve. Enter a value from 1 through 999. For example, to preserve the most recent 15 backups, specify 15 for this value. The default setting is 10.

Parameter/Template Parameter	Required?	Description
`--purge-backup-after` / `purgeBackupAfter`	Optional	Specifies the number of days after which a backup is eligible to be purged. Enter a value from 1 through 999. For example, a value of 45 will purge any backups older than 45 days. However, note that backups older than this number of days may be available because a backup is purged when it is older than the number of days specified in the `--purge-backup-after` / `purgeBackupAfter` parameter and there are more backups remaining than the number of backups specified in the `--recent-backups-to-preserve` / `recentBackupsToPreserve` parameter. The default setting is 30. Enter this value using the ISO 8601 standard. For example, enter P25D to specify 25 days.

JSON Example

1. Generate JSON input for the `okv backup destination-policy create` command:

   ```
   okv backup destination-policy create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_dest_policy_create.json`) and then edit it to create the backup destination policy:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination-policy",
       "action" : "create",
       "options" : {
         "destinationPolicy" : "global_dest_pol",
         "recentBackupsToPreserve" : "15",
         "purgeBackupAfter" : "P45D"
       }
     }
   }
   ```

3. Run the `okv backup destination-policy create` command using the generated JSON file:

   ```
   okv backup destination-policy create --from-json backup_dest_policy_create.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup destination-policy delete

The `okv backup destination-policy delete` command deletes a backup destination policy.

Required Authorization

System Administrator role

Syntax

```
okv backup destination-policy delete --destination_policy policy_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination-policy",
    "action" : "delete",
    "options" : {
      "destinationPolicy" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--destination-policy` / `destinationPolicy`	Required	Specifies the name of the backup destination policy to delete. To find existing backup destination policies, run the `okv backup destination-policy list` command.

JSON Example

1. Generate JSON input for the `okv backup destination-policy delete` command:

   ```
   okv backup destination-policy delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_dest_policy_delete.json`) and then edit it to specify the backup destination policy to delete:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination-policy",
       "action" : "delete",
       "options" : {
         "destinationPolicy" : "global_dest_pol"
       }
     }
   }
   ```

3. Run the `okv backup destination-policy delete` command using the generated JSON file:

```
okv backup destination-policy delete --from-json backup_dest_policy_delete.json
```

Output similar to the following appears:

```
{
  "result" : "Success"
}
```

okv backup destination-policy get

The `okv backup destination-policy get` command retrieves detailed information about a backup destination policy.

Required Authorization

System Administrator role

Syntax

```
okv backup destination-policy get --destination-policy policy_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination-policy",
    "action" : "get",
    "options" : {
      "destinationPolicy" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--destination-policy` / `destinationPolicy`	Required	Specifies the name of the backup destination policy to retrieve. To find existing backup destination policies, run the `okv backup destination-policy list` command.

JSON Example

1. Generate JSON input for the `okv backup destination-policy get` command:

   ```
   okv backup destination-policy get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_dest_policy_get.json`) and then edit it to retrieve the backup destination policy:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination-policy",
       "action" : "get",
   ```

```
    "options" : {
      "destinationPolicy" : "global_dest_pol"
    }
  }
}
```

3. Run the `okv backup destination-policy get` command using the generated JSON file:

   ```
   okv backup destination-policy get --from-json backup_dest_policy_get.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "destinationPolicy" : "GLOBAL_DEST_POL",
       "purgeBackupAfter" : "P45D",
       "recentBackupsToPreserve" : "15"
     }
   }
   ```

okv backup destination-policy list

The `okv backup destination-policy list` command lists existing backup destination policies and their settings.

Required Authorization

System Administrator role

Syntax

```
okv backup destination-policy list
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination-policy",
    "action" : "list"
  }
}
```

Parameters

No parameters

JSON Example

1. Generate JSON input for the `okv backup destination-policy list` command:

   ```
   okv backup destination-policy list --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_dest_policy_list.json`).

3. Run the `okv backup destination-policy list` command using the generated JSON file:

   ```
   okv backup destination-policy list --from-json backup_dest_policy_list.json
   ```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "destinationPolicies" : [
      {
        "destinationPolicy" : "GLOBAL_DEST_POL",
        "recentBackupsToPreserve" : "15",
        "purgeBackupAfter" : "P45D"
      }, {
        "destinationPolicy" : "RAC_DEST_POL",
        "recentBackupsToPreserve" : "5",
        "purgeBackupAfter" : "P60D"
      }
    ],
    "fetchedDestinationPolicyCount" : "2"
  }
}
```

okv backup destination-policy list-purged-backups

The `okv backup destination-policy list-purged-backups` command lists the backups purged by a backup destination policy.

Required Authorization

System Administrator role

Syntax

`okv backup destination-policy list-purged-backups --destination-policy` *`policy_name`* `--destination` *`destination_name`*

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination-policy",
    "action" : "list-purged-backups",
    "options" : {
      "destinationPolicy" : "#VALUE",
      "destination" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--destination-policy` / `destinationPolicy`	Required	Specifies the name of the backup destination policy that you want to list purged backups for. To find existing backup destination policies, run the `okv backup destination-policy list` command.

Parameter/Template Parameter	Required?	Description
`--destination` / `destination`	Optional	Name of the remote backup destination. To find existing backup destination names, run the `okv backup destination list` command. If you omit this setting, then it lists backups purged by a specified backup destination policy on all of the backup destinations.

JSON Example

1. Generate JSON input for the `okv backup destination-policy list-purged-backups` command:

   ```
   okv backup destination-policy list-purged-backups --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_dest_policy_list_purged.json`) and then edit it to include the backup destination policy name and backup destination name:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination-policy",
       "action" : "list-purged-backups",
       "options" : {
         "destinationPolicy" : "global_dest_pol",
         "destination" : "prod_okv_backup_dest"
       }
     }
   }
   ```

3. Run the `okv backup destination-policy list-purged-backups` command using the generated JSON file:

   ```
   okv backup destination-policy list-purged-backups --from-json
   backup_dest_policy_list_purged.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "fetchedPurgedBackupCount" : "2",
       "purgedBackups" : [
         {
           "backupFile" : "okvbackup_periodic_incr_20211030162206",
           "status" : "Purged",
           "purgeTime" : "2021-12-14 19:15:06",
           "destination" : "PROD_OKV_BACKUP_DEST",
           "hostName" : "192.0.2.34",
           "port" : "22",
           "userName" : "psmith",
           "path" : "/opt/okv/backups"
         }, {
           "backupFile" : "okvbackup_onetime_onetime_20211028012206",
           "status" : "Unknown",
   ```

```
            "purgeTime" : "2021-12-12 17:38:06",
            "destination" : "PROD_OKV_BACKUP_DEST",
            "hostName" : "192.0.2.34",
            "port" : "22",
            "userName" : "psmith",
            "path" : "/opt/okv/backups"
          }
        ]
      }
    }
```

okv backup destination-policy update

The `okv backup destination-policy update` command updates a backup destination policy.

Required Authorization

System Administrator role

Syntax

`okv backup destination-policy update --destination-policy` *`policy_name`* `--recent-backups-to-preserve` *`number_of_recent_backups_to_preserve`* `--purge-backup-after` *`number_of_days_after_which_backup_purged`*

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination-policy",
    "action" : "update",
    "options" : {
      "destinationPolicy" : "#VALUE",
      "recentBackupsToPreserve" : "#VALUE",
      "purgeBackupAfter" :  "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--destination-policy` / `destinationPolicy`	Required	Specifies the name of the backup destination policy to update. To find existing backup destination policies, run the `okv backup destination-policy list` command.
`--recent-backups-to-preserve` / `recentBackupsToPreserve`	Optional	The number of the most recent backups to preserve. Enter a value from 1 through 999. For example, to preserve the most recent 15 backups, specify 15 for this value. The default setting is 10.

Parameter/Template Parameter	Required?	Description
`--purge-backup-after` / `purgeBackupAfter`	Optional	Specifies the number of days after which a backup is eligible to be purged. Enter a value from 1 through 999. For example, a value of 45 will purge any backups older than 45 days. However, note that backups older than this number of days may be available because a backup is purged when it is older than the number of days specified in the `--purge-backup-after` / `purgeBackupAfter` parameter and there are more backups remaining than the number of backups specified in the `--recent-backups-to-preserve` / `recentBackupsToPreserve` parameter. The default setting is 30. Enter this value using the ISO 8601 standard. For example, enter P25D to specify 25 days.

JSON Example

1. Generate JSON input for the `okv backup destination-policy update` command:

   ```
   okv backup destination-policy update --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `backup_dest_policy_update.json`) and then edit it to include the backup destination policy:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination-policy",
       "action" : "update",
       "options" : {
         "destinationPolicy" : "global_dest_pol",
         "recentBackupsToPreserve" : "20",
         "purgeBackupAfter" : "P60D"
       }
     }
   }
   ```

3. Run the `okv backup destination-policy update` command using the generated JSON file:

   ```
   okv backup destination-policy update --from-json backup_dest_policy_update.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup history list

The `okv backup history list` command lists the details of a backup history, such as runtime errors, whether the backup completed, and start and end times.

Required Authorization

System Administrator role

Syntax

```
okv backup history list --max number_of_backup_records
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "history",
    "action" : "list",
    "options" : {
      "max" : "#VALUE",
      "name" : "#VALUE",
      "type" : "#ONE-TIME|PERIODIC|PERIODIC-FULL",
      "destination" : "#LOCAL|VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--max` / `max`	Required	The maximum number of the most recent complete backup information records that should be returned
`--type` / `type`	Optional	The endpoint platform. The allowed values are: • `ONE-TIME` • `PERIODIC` • `PERIODIC-FULL` You can also specify a combination of comma separated values. The filter is applied to each value independently, and the combined results returned effectively supports an OR operation.
`--destination` / `destination`	Optional	Backup destination name. You can also specify multiple comma separated values of backup destination names. The filter is applied to each backup destination name independently and the combined results returned effectively supports an OR operation.
`--name` / `name`	Optional	Backup name. You can also specify multiple backup destination names separated by a comma. The filter is applied to each backup destination name independently and the combined results returned effectively supports an OR operation.

JSON Example

1. Generate JSON input for the `okv backup history list` command:

   ```
   okv backup history list --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_history_list.json`) and then edit it to generate the backup history list:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "history",
       "action" : "list",
       "options" : {
         "max" : "10"
       }
     }
   }
   ```

3. Run the `okv backup history list` command using the generated JSON file:

   ```
   okv backup history list --from-json bkup_history_list.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "backups": [
     {
       "backupTime": "2020-11-30 03:59:36",
       "destination": "PROD_OKV_BACKUP_DEST",
       "interval": "0:1:0",
       "lastFullBackupTime": "2020-11-30 01:27:31",
       "name": "OKV_BACKUP_HOURLY",
       "runError": " ",
       "runIndex": "2",
       "scheduleTime": "2020-11-30 01:09:56",
       "startTime": "2020-11-30 03:42:09",
       "status": "DONE",
       "type": "PERIODIC"
     },
     {
     "backupTime": "2020-11-30 02:13:36",
     "destination": "LOCAL",
     "interval": "0:0:0",
     "lastFullBackupTime": "2020-11-30 02:13:36",
     "name": "LOCAL_BACKUP",
     "runError": " ",
     "runIndex": "1",
     "scheduleTime": "2020-11-30 01:40:28",
     "startTime": "2020-11-30 02:00:02",
     "status": "DONE",
     "type": "ONE-TIME"
     },
     {
     "backupTime": "2020-11-30 01:27:31",
     "destination": "PROD_OKV_BACKUP_DEST",
     "interval": "0:1:0",
     "lastFullBackupTime": "2020-11-30 01:27:31",
   ```

```
    "name": "OKV_BACKUP_HOURLY",
    "runError": " ",
    "runIndex": "1",
    "scheduleTime": "2020-11-30 01:09:56",
    "startTime": "2020-11-30 01:10:00",
    "status": "DONE",
    "type": "PERIODIC"
    }
    ]
  }
```

okv backup restore start

The `okv backup restore start` command starts the restore process of an Oracle Key Vault backup.

This command will require the use of the Oracle Key Vault recovery passphrase.

Required Authorization

System Administrator role

After you begin the restore operation, you can check its status by running the `okv backup restore status` command.

Syntax

`okv backup restore start --name` *`backup_file_name`* `--destination` *`destination_name`*

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "restore",
    "action" : "start",
    "options" : {
      "name" : "#VALUE",
      "destination" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name for the backup file. To find existing scheduled backups, run the `okv backup schedule list` command.
`--destination` / `destination`	Required	Name of the backup destination. For a listing of existing names, run the `okv backup destination list` command.

JSON Example

1. Generate JSON input for the `okv backup restore start` command:

```
okv backup restore start --generate-json-input
```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_restore_start.json`) and then edit it to restore the backup. In the following example, the `passphrase` is not specified as the user will be prompted for it:

```
{
  "service" : {
    "category" : "backup",
    "resource" : "restore",
    "action" : "start",
    "options" : {
      "name" : "okvbackup_onetime_onetime_20210118175804",
      "destination" : "prod_okv_backup_dest"
    }
  }
}
```

3. Run the `okv backup restore start` command using the generated JSON file:

```
okv backup restore start --from-json bkup_restore_start.json
```

You will be prompted to enter the recovery passphrase, which was valid at the time of the backup that is being restored. After you enter the passphrase, output similar to the following appears:

```
{
  "result" : "Success"
}
```

okv backup restore status

The `okv backup restore status` command checks the status of the Oracle Key Vault backup restore operation.

Required Authorization

System Administrator role

Syntax

```
okv backup restore status
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "restore",
    "action" : "status"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv backup restore status` command:

   ```
   okv backup restore status --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.
2. Save the generated input to a file (for example, `bkup_restore_status.json`).
3. Run the `okv backup restore status` command using the generated JSON file:

   ```
   okv backup restore status --from-json bkup_restore_status.json
   ```

   If the restore process is ongoing, then output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "destination" : "LOCAL",
       "status" : "ONGOING",
       "time" : "2020-12-18 11:49:30"
     }
   }
   ```

   If the restore process is complete, then output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "destination" : "LOCAL",
       "status" : "DONE",
       "time" : "2020-12-18 11:52:02"
     }
   }
   ```

okv backup schedule create

The `okv backup schedule create` command creates a backup schedule job.

Required Authorization

System Administrator role

Syntax

```
okv backup schedule create
--name backup_schedule_name
--start-time start_time
--destination LOCAL|VALUE
--type ONE-TIME|PERIODIC|PERIODIC-FULL
--interval timing
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "create",
    "options" : {
```

```
      "name" : "#VALUE",
      "startTime" : "#VALUE",
      "type" : "#ONE-TIME|PERIODIC|PERIODIC-FULL",
      "destination" : "#LOCAL|VALUE",
      "interval" : {
        "days" : "#0-99",
        "hours" : "#0-23",
        "mins" : "#0-59"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name for the backup schedule job. To find existing scheduled backups, run the `okv backup schedule list` command.

`--start-time` / `startTime`	Required	Time to begin the scheduled backup.
		You can use different ways to set the date and time. Examples are as follows:
		`"startTime" : "now"` *`--starts immediately`*
		`"startTime" : "now+PT10M"` *`--starts 10 minutes from now`*
		`"startTime" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"startTime" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		To display the time in UTC format, use the Linux `date` command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`

`--destination` / `destination`	Required	Type or name of backup destination. Enter one of the following values: • `LOCAL` • For a remote destination, enter its name. To find existing names, run the `okv backup destination list` command.
`--type` / `type`	Optional	Enter one of the following values: • `ONE-TIME` (Default value) • `PERIODIC` • `PERIODIC-FULL`
`--interval` / `interval`	Required for periodic backups	Enter the days, hours, and minutes in between the backups, using the following format: *`days`*`:`*`hours`*`:`*`minutes`*

JSON Example

1. Generate JSON input for the `okv backup schedule create` command:

   ```
   okv backup schedule create --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_sched_create.json`) and then edit it to create the backup schedule:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "create",
       "options" : {
         "name" : "okv_backup_hourly",
         "startTime" : "2020-11-30 01:10:00",
         "type" : "PERIODIC",
         "destination" : "prod_okv_backup_dest",
         "interval" : {
           "days" : "0",
           "hours" : "1",
           "mins" : "0"
         }
       }
     }
   }
   ```

3. Run the `okv backup schedule create` command using the generated JSON file:

   ```
   okv backup schedule create --from-json bkup_sched_create.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

okv backup schedule delete

The `okv backup schedule delete` command deletes scheduled backup job.

Required Authorization

System Administrator role

Syntax

```
okv backup schedule delete --name backup_schedule_name
```

JSON Input File Template

```
{
  "service" : {
```

```
    "category" : "backup",
    "resource" : "schedule",
    "action" : "delete",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup schedule job. To find existing scheduled backups, run the `okv backup schedule list` command. If you need more details about the backup that you want to delete, then run `okv backup schedule get`.

JSON Example

1. Generate JSON input for the `okv backup schedule delete` command:

   ```
   okv backup schedule delete --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_sched_del.json`) and then edit it to delete the backup schedule:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "delete",
       "options" : {
         "name" : "okv_backup_hourly"
       }
     }
   }
   ```

3. Run the `okv backup schedule delete` command using the generated JSON file:

   ```
   okv backup schedule delete --from-json bkup_sched_del.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup schedule get

The `okv backup schedule get` command retrieves detailed information about a scheduled Oracle Key Vault server backup.

The `okv backup schedule get` command returns the following values:

- Type (for example, `periodically @ 1 days 0 hrs 0 mins`)

- Destination (for example, `local` or `remote-dest`)
- State (`ACTIVE`, `ONGOING`, `PAUSED`, `DONE`)
- Last run error
- Schedule time
- Start time
- Last backup time
- Last full backup time
- Run count (for periodic backups)

Required Authorization

System Administrator role

Syntax

```
okv backup schedule get --name backup_schedule_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "get",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup schedule job. To find existing scheduled backups, run the `okv backup schedule list` command.

JSON Example

1. Generate JSON input for the `okv backup schedule get` command:

   ```
   okv backup schedule get --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_sched_get.json`) and then edit it to get the backup schedule:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "get",
       "options" : {
   ```

```
      "name" : "okv_backup_hourly"
    }
  }
}
```

3. Run the `okv backup schedule get` command using the generated JSON file:

   ```
   okv backup schedule get --from-json bkup_sched_get.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" : {
       "destination" : "PROD_OKV_BACKUP_DEST",
       "interval" : "0:1:0",
       "name" : "OKV_BACKUP_HOURLY",
       "runIndex" : "0",
       "scheduleTime" : "2021-01-16 18:37:15",
       "startTime" : "2021-03-31 18:36:00",
       "status" : "ACTIVE",
       "type" : "PERIODIC"
     }
   }
   ```

okv backup schedule list

The `okv backup schedule list` command displays a listing of the currently scheduled Oracle Key Vault server backups.

Required Authorization

System Administrator role

Syntax

```
okv backup schedule list
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "list"
  }
}
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv backup schedule list` command:

   ```
   okv backup schedule list --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_sched_list.json`).

3. Run the `okv backup schedule list` command using the generated JSON file:

```
okv backup schedule list --from-json bkup_sched_list.json.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "schedules" : [
      "OKV_BACKUP_HOURLY",
      "LOCAL_BACKUP"
    ]
  }
}
```

okv backup schedule pause

The `okv backup schedule pause` command pauses a scheduled Oracle Key Vault server backup.

Required Authorization

System Administrator role

Syntax

```
okv backup schedule pause --name backup_schedule_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "pause",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup schedule job. To find existing scheduled backups, run the `okv backup schedule list` command. If you need more details about the backup that you want to pause, then run `okv backup schedule get`.

JSON Example

1. Generate JSON input for the `okv backup schedule pause` command:

```
okv backup schedule pause --generate-json-input
```

The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup-sched-pause.json`) and then edit it so that you can pause the backup schedule:

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "pause",
    "options" : {
      "name" : "okv_backup_hourly"
    }
  }
}
```

3. Run the `okv backup schedule pause` command using the generated JSON file:

```
okv backup schedule pause --from-json bkup-sched-pause.json
```

Output similar to the following appears:

```
{
  "result" : "Success"
}
```

okv backup schedule resume

The `okv backup schedule resume` command resumes a paused Oracle Key Vault backup job.

Required Authorization

System Administrator role

Syntax

```
okv backup schedule resume --name backup_schedule_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "resume",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup schedule job. To find existing scheduled backups, run the `okv backup schedule list` command. If you need more details about the backup that you want to resume, then run `okv backup schedule get`.

JSON Example

1. Generate JSON input for the `okv backup schedule resume` command:

   ```
   okv backup schedule resume --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_sched_resume.json`) and then edit it to resume the paused backup schedule:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "resume",
       "options" : {
         "name" : "okv_backup_hourly"
       }
     }
   }
   ```

3. Run the `okv backup schedule resume` command using the generated JSON file:

   ```
   okv backup schedule resume --from-json bkup_sched_resume.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup schedule run-now

The `okv backup schedule run-now` command runs the backup schedule job immediately.

Required Authorization

System Administrator role

Syntax

```
okv backup schedule run-now --name backup_schedule_name
```

JSON Input File Template

```
{
  "service" : {
```

```
    "category" : "backup",
    "resource" : "schedule",
    "action" : "run-now",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

Table 10-1 Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name for the backup schedule job that is required to run immediately.

JSON Example

1. GenerateJSON input for the `okv backup schedule run-now` command:

   ```
   okv backup schedule run-now --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_sched_run_now.json`) and then edit it to immediately run a backup associated with a backup schedule:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "run-now",
       "options" : {
         "name" : "okv_backup_hourly"
       }
     }
   }
   ```

3. Run the `okv backup schedule run-now` command using the generated JSON file:

   ```
   okv backup schedule run-now --from-json bkup_sched_run_now.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

okv backup schedule update

The `okv backup schedule update` command updates a currently scheduled backup.

Required Authorization

System Administrator role

Syntax

```
okv backup schedule update --name backup_schedule_name --start-time start_time --
interval timing
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "update",
    "options" : {
      "name" : "#VALUE",
      "startTime" : "#VALUE",
      "interval" : {
        "days" : "#0-99",
        "hours" : "#0-23",
        "mins" : "#0-59"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup schedule job. To find the names of existing scheduled backups, run the `okv backup schedule list` command. To find the details of the backup that you want to update, run `okv backup schedule get`.

`--start-time` / `startTime`	Optional	Time to begin the scheduled backup.
		You can use different ways to set the date and time. Examples are as follows:
		`"startTime" : "now"` *`--starts immediately`*
		`"startTime" : "now+PT10M"` *`--starts 10 minutes from now`*
		`"startTime" : "2021-12-20 10:30:00"` *`--starts at this date and time`*
		`"startTime" : "2021-12-20 10:30:00+PT10M"` *`--starts 10 minutes after this date and time`*
		To display the time in UTC format, use the Linux `date` command. For example:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`

`--interval` / `interval`	Required for periodic backups	Enter the days, hours, and minutes in between the backups, using the following format: *`days`*`:`*`hours`*`:`*`minutes`*

JSON Example

1. Generate JSON input for the `okv backup schedule update` command:

   ```
   okv backup schedule update --generate-json-input
   ```

   The generated JSON input is displayed as shown in the earlier section **JSON Input File Template**.

2. Save the generated input to a file (for example, `bkup_sched_update.json`) and then edit it to update the backup schedule:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "update",
       "options" : {
         "name" : "okv_backup_hourly",
         "startTime" : "2020-12-20 18:00:00",
         "interval" : {
           "days" : "0",
           "hours" : "12",
           "mins" : "0"
         }
       }
     }
   }
   ```

3. Run the `okv backup schedule update` command using the generated JSON file:

   ```
   okv backup schedule update --from-json bkup_sched_update.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success"
   }
   ```

Related Topics

- How to Set the Date and Time in RESTful Services Utility Commands
  You specify the date or timestamp, and duration using the supported formats.

11

Logging and Error Reporting

Logging and error reporting can help with troubleshooting issues that may arise.

- Configuring Logging
  You can configure a variety of ways to handle logging, including specifying a range of different logging levels.
- Error Reporting
  The RESTful Service utility has robust error reporting to debug in order to run RESTful service commands quickly and successfully.

Configuring Logging

You can configure a variety of ways to handle logging, including specifying a range of different logging levels.

- About Configuring Logging
  Using RESTful service logging configuration, you generate diagnostic logging information to help with troubleshooting issues that may arise.
- Log Property File Parameters
  The log property file has parameters to control aspects such as handlers, log levels, output file names, and so on.
- Example: Logging File
  Oracle Key Vault RESTful services logging files are based on the `java.util.logging` Java logging utility.

About Configuring Logging

Using RESTful service logging configuration, you generate diagnostic logging information to help with troubleshooting issues that may arise.

Oracle Key Vault RESTful service logging is based on the customized Java logging utility, `java.util.logging`.

The RESTful service logging configuration is specified in a log property file. In the log property file, you can accomplish the following:

- Specify whether log messages are sent to the console, to a file or to both
- Specify the file names where log messages are written
- Specify the level of detail in the log messages.
- Specify the log formatter to use. You can choose XML or regular text format. Each log entry shows class, object, and method, along with the time when the log entry is generated.

To specify the log property file in the Oracle Key Vault RESTful services configuration file (`okvrestcli.ini`), you use the `log_property` parameter.

Log Property File Parameters

The log property file has parameters to control aspects such as handlers, log levels, output file names, and so on.

Table 11-1 describes the Oracle Key Vault RESTful logging property file parameters.

Table 11-1 Log Property File Parameters

Logging Property	Description
`handlers`	A comma-delimited list of handler classes to output log messages. Available handlers are `java.util.logging.FileHandler` and `java.util.logging.ConsoleHandler`. The `FileHandler` can either write to a specified file, or it can write to a rotating set of files. Handlers can have both `ConsoleHandler` and `FileHandler` separated by a comma (`,`).
`.level`	Sets the log level for all `FileHandler` instances. Levels are as follows: • `ALL` indicates that all messages should be logged. • `FINE` is a message level providing tracing information. • `FINER` indicates a fairly detailed tracing message. • `FINEST` indicates a highly detailed tracing message. • `INFO` is a message level for informational messages. • `OFF` is a special level that can be used to turn off logging. • `SEVERE` indicates a serious failure. • `WARNING` indicates a potential problem.
`java.util.logging.FileHandler.level`	See the description of `.level` for descriptions of these levels that you can use in `java.util.logging.FileHandler.level`
`java.util.logging.FileHandler.pattern`	Specifies a pattern for generating the output file name. A pattern consists of a string that includes the following special components that will be replaced at runtime: • `/` is the local path name separator. • `%t` is the system temporary directory. • `%h` is the value of the `user.home` system property. • `%g` is the generation number to distinguish rotated logs. If no `%g` field has been specified and the file count is greater than one, then the generation number will be added to the end of the generated file name, after a dot. • `%u` is a unique number to resolve conflicts between simultaneous Java processes. • `%%` translates to a single percent sign `%`.

Table 11-1 (Cont.) Log Property File Parameters

Logging Property	Description
`java.util.logging.FileHandler.limit`	The maximum size of the file, in bytes. If this is 0, then there is no limit. The default is 200000. Logs larger than the specified limit roll over to the next log file.
`java.util.logging.FileHandler.count`	The number of log files to use in the log file rotation. The default is 5.
`java.util.logging.FileHandler.formatter`	Specifies the name of a `Formatter` class to use. To generate the log entries in the XML format, use `java.util.logging.XMLFormatter`. To generate the log entries in the plain text, use `com.oracle.okv.rest.log.OkvFormatter`.
`java.util.logging.ConsoleHandler.level`	Sets the default log level for all `ConsoleHandler` instances. See the description of the `java.util.logging.FileHandler.level` for descriptions of these levels.
`java.util.logging.ConsoleHandler.formatter`	Specifies the name of a `Formatter` class to use. To generate the log entries in the XML format, use `java.util.logging.XMLFormatter`. To generate the log entries in the plain text, use `com.oracle.okv.rest.log.OkvFormatter`.

Example: Logging File

Oracle Key Vault RESTful services logging files are based on the `java.util.logging` Java logging utility.

Example 11-1 shows a logging file that uses the `INFO` logging level.

Example 11-1 Logging File

```
handlers= java.util.logging.FileHandler
.level=INFO

# default file output is in log directory.
java.util.logging.FileHandler.pattern = /usr/local/okv/okvrestcli.log
java.util.logging.FileHandler.limit = 200000
java.util.logging.FileHandler.count = 4
java.util.logging.FileHandler.level = INFO
java.util.logging.FileHandler.formatter = com.oracle.okv.rest.log.OkvFormatter
```

Error Reporting

The RESTful Service utility has robust error reporting to debug in order to run RESTful service commands quickly and successfully.

About Error Reporting

Depending upon the logging configuration, Oracle Key Vault may write additional information about the failure to the log file.

The specific error will be reported, with suggestions for corrective actions. Error reporting is common to all REST commands.

The first thing to do when investigating a command failure is to look into the log file. If you have not created a custom log file in a location of your choice, then you can look at the default log file, `okvrestcli.log` in the `conf` directory

To see all the messages from the Oracle Key Vault server during command execution, you can set the appropriate logging level, log file name, and the log file location in the configuration file.

The RESTful service utility reports errors such as the failure to locate a file or an environment variable like `JAVA_HOME`, incorrect command syntax, and incorrect passwords.

Command Line Error Reporting

Error reporting captures both faulty actions, such as incorrect passwords, and successful command executions.

Example: Error: Incorrect Password

```
okv admin endpoint update --user psmith --endpoint hr_db_ep --description 'HR DB
Endpoint'
Password: password
{
  "result" : "Failure",
  "message" : "Invalid username or password. Try again after 5 seconds."
}
```

Example: Successful Service Command Execution

```
okv admin endpoint update --user psmith -endpoint hr_db_ep --description 'HR DB Endpoint'
Password: password
{
  "result" : "Success"
}
```

Example: Log File Entry

In addition to the helpful error and usage messages, an entry for the action is logged in the log file with the date.

```
Thu Oct 29 15:50:19 PDT 2020::com.oracle.okv.rest.cli.okv::main::1::[backup, history,
list, --max, 5]
Thu Oct 29 15:50:19 PDT
2020::com.oracle.okv.rest.cli.backup.BackupProcessManager::<init>::1::https://
10.240.112.193:5695/okv/cloud/api
Thu Oct 29 15:50:19 PDT
```

```
2020::com.oracle.okv.rest.cli.backup.BackupProcessManager::<init>::1::/scratch/dopark/
demo/EP1/ssl
Thu Oct 29 15:50:19 PDT
2020::com.oracle.okv.rest.cli.backup.BackupOptionsProcessor::takeOption::1::BackupOptionB
ean
[name=null, startTime=null, destination=null, type=null, max=5, interval=null,
passphrase=null,
transferMethod=null, hostName=null, port=null, path=null, userName=null,
authenticationMethod=null,
psd=null, cluster=false]
```

A

Oracle Key Vault RESTful Services Utility Commands Change History

The Oracle Key Vault RESTful services utility commands changed dramatically starting in release 21.

- Oracle Key Vault Pre-Release 21.1 Commands Comparison
  The Oracle Key Vault pre-release 21.1 commands and enable you to work with endoints, endpoint groups, wallets, keys, and security objects.
- Commands New with Oracle Key Vault Release 21.1
  The commands that are new with Oracle Key Vault release 21.1 enable you to work with monitoring operations, and backup and restore operations.

Oracle Key Vault Pre-Release 21.1 Commands Comparison

The Oracle Key Vault pre-release 21.1 commands and enable you to work with endoints, endpoint groups, wallets, keys, and security objects.

Table A-1 describes how the Oracle Key Vault commands changed in release 21.1.

Table A-1 Changes to Oracle Key Vault RESTful Commands

Pre-Release 21 Command	Release 21 Command Equivalent
`activate`	`okv managed-object object activate`
`add_attr`	`okv managed-object attribute add`
`add_custom_attr`	`okv managed-object custom-attribute add`
`add_epg_member`	`okv manage-access endpoint-group add-endpoint`
`add_member`	`okv managed-object wallet add-member`
`add_wallet_access_ep`	An option in `okv manage-access wallet add-access`
`add_wallet_access_epg`	An option in `okv manage-access wallet add-access`
`check_object_status`	Covered by the following commands: • `okv admin endpoint check-status` • `okv manage-access endpoint-group check-status` • `okv manage-access wallet check-status`
`create_endpoint`	`okv admin endpoint create`
`create_endpoint_group`	`okv manage-access endpoint-group create`
`create_key`	`okv managed-object key create`
`create_unique_endpoint`	As an option in `okv admin endpoint create`

Table A-1 (Cont.) Changes to Oracle Key Vault RESTful Commands

Pre-Release 21 Command	Release 21 Command Equivalent
`create_unique_endpoint_group`	As an option in `okv manage-access endpoint-group create`
`create_unique_wallet`	As an option in `okv manage-access wallet create`
`create_wallet`	`okv manage-access wallet create`
`del_attr`	`okv managed-object attribute delete`
`del_custom_attr`	`okv managed-object custom-attribute delete`
`del_member`	`okv managed-object wallet delete-member`
`delete_endpoint`	`okv admin endpoint delete`
`delete_endpoint_group`	`okv manage-access endpoint-group delete`
`delete_wallet`	`okv manage-access wallet delete`
`destroy`	`okv managed-object object destroy`
`download`	`okv admin endpoint download`
`drop_epg_member`	`okv manage-access endpoint-group remove-endpoint`
`drop_wallet_access_ep`	An option in `okv manage-access wallet remove-access`
`drop_wallet_access_epg`	An option in `okv manage-access wallet remove-access`
`get_attr`	`okv managed-object attribute get`
`get_cert`	`okv managed-object certificate get`
`get_default_wallet`	`okv manage-access wallet get-default`
`get_enrollment_token`	`okv admin endpoint get-enrollment-token`
`get_key`	`okv managed-object key get`
`get_opaque`	`okv managed-object opaque get`
`get_secret`	`okv managed-object secret get`
`get_system_info`	Covered by the following commands: • `okv cluster info get` • `okv primary-standby info get` • `okv server info get`
`get_wallets`	`okv manage-access wallet list-endpoint-wallets`
`list_attr`	`okv managed-object attribute list`
`list_wallet`	`okv managed-object wallet list`
`locate`	`okv managed-object object locate`
`mod_attr`	`okv managed-object attribute modify`
`mod_custom_attr`	`okv managed-object custom-attribute modify`
`modify_endpoint_desc`	An option in `okv admin endpoint update`

Table A-1 (Cont.) Changes to Oracle Key Vault RESTful Commands

Pre-Release 21 Command	Release 21 Command Equivalent
`modify_endpoint_email`	An option in `okv admin endpoint update`
`modify_endpoint_group_desc`	An option in `okv manage-access endpoint-group update`
`modify_endpoint_group_name`	An option in `okv manage-access endpoint-group update`
`modify_endpoint_name`	An option in `okv admin endpoint update`
`modify_endpoint_platform`	An option in `okv admin endpoint update`
`modify_endpoint_type`	An option in `okv admin endpoint update`
`modify_wallet_access_ep`	An option in `okv manage-access wallet update-access`
`modify_wallet_access_epg`	An option in `okv manage-access wallet update-access`
`modify_wallet_desc`	An option in `okv manage-access wallet update`
`modify_wallet_name`	An option in `okv manage-access wallet update`
`provision`	`okv admin endpoint provision`
`query`	`okv managed-object object query`
`re_enroll`	`okv admin endpoint re-enroll`
`re_enroll_all`	`okv admin endpoint re-enroll-all`
`reg_cert`	`okv managed-object certificate register`
`reg_key`	`okv managed-object key register`
`reg_opaque`	`okv managed-object opaque register`
`reg_secret`	`okv managed-object secret register`
`revoke`	`okv managed-object object revoke`
`set_default_wallet`	`okv manage-access wallet set-default`

Commands New with Oracle Key Vault Release 21.1

The commands that are new with Oracle Key Vault release 21.1 enable you to work with monitoring operations, and backup and restore operations.

The following commands that cover RESTful services functionality were not available prior to Oracle Key Vault 21.1:

- `okv admin client-wallet add`
- `okv admin client-wallet delete`
- `okv admin client-wallet list`
- `okv admin client-wallet update`
- `okv backup destination create`
- `okv backup destination delete`

- `okv backup destination get`
- `okv backup destination get-public-key`
- `okv backup destination list`
- `okv backup destination list-backups`
- `okv backup destination reset-host-key`
- `okv backup destination update`
- `okv backup history list`
- `okv backup restore start`
- `okv backup restore status`
- `okv backup schedule create`
- `okv backup schedule delete`
- `okv backup schedule get`
- `okv backup schedule list`
- `okv backup schedule pause`
- `okv backup schedule resume`
- `okv backup schedule update`
- `okv cluster status get`
- `okv managed-object attribute get-all`
- `okv primary-standby status get`
- `okv server status get`

Index

G

H

I

J

K

L

M

O

P

R

S

T

U

V

W