Oracle® Key Vault

RESTful Services Administrator's Guide

Release 21.3
F45657-03
May 2022

Oracle Key Vault RESTful Services Administrator's Guide, Release 21.3

F45657-03

Primary Author: Monika Sharma

Contributors: 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

Contents

3 Administration Commands

4 Access Management Commands

5 Security Object 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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

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 21.3
  Oracle Key Vault release 21.3 introduces one new feature that affects this guide.
- Changes for Oracle Key Vault Release 21.2
  Oracle Key Vault release 21.2 introduces one new feature that affects this guide.

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.

Changes for Oracle Key Vault Release 21.2

Oracle Key Vault release 21.2 introduces one new feature that affects this guide.

- New and Changed RESTful Services Utility Commands
  In Oracle Key Vault release 21.2, several new and changed `okv managed-object` RESTful services utility commands are available.

New and Changed RESTful Services Utility Commands

In Oracle Key Vault release 21.2, several new and changed `okv managed-object` RESTful services utility commands are available.

The new `okv managed-object` RESTful services commands, which add support for `get` and `register` operations for certificate requests, private keys, and public keys, are as follows:

- `okv managed-object certificate-request get`
- `okv managed-object certificate-request register`
- `okv managed-object private-key get`
- `okv managed-object private-key register`
- `okv managed-object public-key get`
- `okv managed-object public-key register`

The changed `okv managed-object` RESTful services commands are as follows:

- `okv managed-object certificate register`
- `okv managed-object object locate`

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.

1.1 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 execute 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 execute 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.

1.2 General Process for Using Oracle Key Vault RESTful Services

After you enable the RESTful services, in most 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 execute 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 execute the commands individually, using different methods. In most cases, the RESTful services utility commands support JSON formatting. Executing RESTful Services Utility Commands Using the Command Line explains how to execute these commands.

1.3 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 execute RESTful services utility commands.
- Executing Oracle Key Vault RESTful Services Utility Commands
  Oracle Key Vault provides a variety of ways to execute 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.
- Using RESTful Services with LDAP Users
  Both regular Oracle Key Vault administrators and properly authorized LDAP users can log in to a server to execute Oracle Key Vault RESTful services utility commands.

2.1 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.

2.1.1 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 have enabled 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.

2.1.1.1 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 1.7.0.21 or later (the Java Runtime Environment (JRE) is provided with Oracle Database release 12.2, so you do not need to install the JRE if you already have Oracle Database release 12.2 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 a JRE version 1.7.0.21 or later, and then set `JAVA_HOME` and `PATH` appropriately. OpenJDK is not supported.

2.1.1.2 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**.

2.1.1.3 Step 3: Enable RESTful Services

After you have enabled 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. Check the box to the right of **Enable**.
4. Click **Save**.

2.1.1.4 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
```

2.1.1.5 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 execute RESTful services utility commands.

2.1.2 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**.

2.2 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 execute 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.

2.2.1 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 execute Oracle Key Vault RESTful services utility commands.

- okvrestcli.ini Configuration Parameters
  The `okvrestcli.ini` parameters cover settings such as the name and password 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 execute 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.

2.2.1.1 About the okvrestcli.ini Configuration File

The `okvrestcli.ini` file enables you to configure commonly used settings when you execute 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 execute 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 execute 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.

2.2.1.2 okvrestcli.ini Configuration Parameters

The `okvrestcli.ini` parameters cover settings such as the name and password 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 execute 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 execute managed-object RESTful services utility commands, which must always be executed 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 execute the command.

  Oracle Key Vault does not use the `user` parameter when the RESTful services utility commands for the managed-object category are executed. These commands are always executed 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 executed. These commands are always executed with the identity of an endpoint that is set with the `okv_client_config` parameter.
- `password`: Specifies the password of the user executing the RESTful services utility commands. If `client_wallet` is specified, then the `password` parameter is not required. If both `client_wallet` and `password` parameters are specified, then the `password` parameter takes precedence over `client_wallet`.

  Oracle Key Vault does not use the `password` parameter when the RESTful services utility commands for the managed-object category are executed. 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.

2.2.1.3 [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 execute 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 execute an Oracle Key Vault RESTful service command, the configuration parameter values are determined on the basis of an order of precedence.

2.2.1.4 Precedence Order of okvrestcli.ini Parameters

When you execute 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 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
  ```

2.2.1.5 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 execute 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 and password 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 execute an Oracle Key Vault RESTful service command, the configuration parameter values are determined on the basis of an order of precedence.

2.2.2 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 execute 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
```

2.3 Executing Oracle Key Vault RESTful Services Utility Commands

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

- RESTful Services Utility Command Syntax
  The RESTful services utility command syntax operates using the `okv` command.
- Ways of Executing RESTful Services Utility Commands
  You can execute 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.

2.3.1 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 the 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 execute 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.

2.3.2 Ways of Executing RESTful Services Utility Commands

You can execute 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.

- Executing RESTful Services Utility Commands Using the Command Line
  You execute the RESTful services utility commands from the command line by specifying all command-specific parameters in the command line.
- Executing 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.
- Naming Conventions for Parameters Executed 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.

2.3.2.1 Executing RESTful Services Utility Commands Using the Command Line

You execute 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 Executed 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.

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

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

1. Generate JSON input designed specifically for the command, by executing 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,WRAP_KEY,UNWRAP_KEY|
   EXPORT,DERIVE_KEY,GENERATE_CRYPTOGRAM,VALIDATE_CRYPTOGRAM,TRANSLATE_ENCRYP
   T,TRANSLATE_DECRYPT,TRANSLATE_WRAP,TRANSLATE_UNWRAP",
         "wallet" : "#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"
    }
  }
}
```

3. To perform the action, execute 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 execute 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 key_create.json --
     length 128
     ```

     Overriding command parameters in the command line allows use of the same JSON file for executing 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" : {
           "attributes" : {
             "contactInfo" : "pfitch@example.com",
             "deactivationDate" : "2024-12-31 09:00:00",
             "name" : "PROD-HRDB-MKEY",
             "protectStopDate" : "2024-09-30  09:00:00"
           }
         }
       }
     }
     ```

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

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

Related Topics

- Naming Conventions for Parameters Executed 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.

2.3.2.3 Naming Conventions for Parameters Executed 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.

2.3.3 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 execute 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`.

- Ensure that you have set the `JAVA_HOME` environment variable to a Java Runtime Environment (JRE) installation with the minimum version 1.7.0.21. For Oracle Database release 12.2.0.1 and later, you can use the Java installation under `$ORACLE_HOME`. OpenJDK is not supported.

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 or the host name 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. Execute 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
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 execute the scripts:

    - Execute the `primary-run-me.sh` script on single-instance databases or the first Oracle RAC instance. For an Oracle Data Guard environment, execute the script on the lead node of the primary Oracle RAC database.
    - Execute 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.

2.4 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. If you are in a multi-master cluster environment that has any nodes that have not yet been upgraded to Oracle Key Vault release 18.5 or later, then use a maximum of 24 bytes for the object name.
- The names of users, user groups, endpoints, and endpoint groups are not case sensitive. For example, `pfitch` and `PFITCH` 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.

2.5 Using RESTful Services with LDAP Users

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

When an LDAP user executes the Oracle Key Vault RESTful services utility commands, Oracle Key Vault first authenticates the user before command is executed. 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.
- Endpoint Management Commands
  The endpoint management commands enable you to perform endpoint-related tasks such as creating or provisioning endpoints.

3.1 Client Wallet Management Commands

You can use the client wallet management commands to manage client wallets that store user credentials.

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

3.1.1 okv admin client-wallet add Command

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

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 input appears as follows:

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

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

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

3. Execute 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"
}
```

3.1.2 okv admin client-wallet delete Command

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "admin",
       "resource" : "client-wallet",
       "action" : "delete",
       "options" : {
   ```

```
      "clientWallet" : "#VALUE",
      "walletUser" : "#VALUE"
    }
  }
}
```

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

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

3. Execute 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"
}
```

3.1.3 okv admin client-wallet list Command

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 input appears as follows:

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

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. Execute 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", "pfitch" ]
     }
   }
   ```

3.1.4 okv admin client-wallet update Command

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 input appears as follows:

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

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

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

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

3.2 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 Command
  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 Command
  The `okv admin endpoint create` command adds a new endpoint to Oracle Key Vault.
- okv admin endpoint delete Command
  The `okv admin endpoint delete` command removes an endpoint from Oracle Key Vault.
- okv admin endpoint download Command
  The `okv admin endpoint download` command downloads the endpoint software (`okvclient.jar`) to the specified directory.
- okv admin endpoint get-enrollment-token Command
  The `okv admin endpoint get-enrollment-token` command retrieves an enrollment token for a registered endpoint.

- okv admin endpoint provision Command
  The `okv admin endpoint provision` command downloads and installs the endpoint software in the specified directory.
- okv admin endpoint re-enroll Command
  The `okv admin endpoint re-enroll` command re-enrolls a previously enrolled endpoint.
- okv admin endpoint re-enroll-all Command
  The `okv admin endpoint re-enroll-all` command re-enrolls all previously enrolled endpoints.
- okv admin endpoint update Command
  The `okv admin endpoint update` command updates the settings of an endpoint.

3.2.1 okv admin endpoint check-status Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page. To find the locator ID in the Oracle Key Vault management console, select the **Cluster** tab and then in the left navigation bar, select **Conflict Resolution**. In the Keys, Secrets & Objects table, check the **Unique Identifier** column.

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 input appears as follows:

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

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. Execute 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.

3.2.2 okv admin endpoint create Command

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

Required Authorization

System Administrator role or the Create Endpoint system privilege

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

Syntax

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

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",
      "type" : "#ORACLE_DB|ORACLE_NON_DB|ORACLE_ACFS|MYSQL_DB|OTHER",
      "subgroup" : "#VALUE|NO SUBGROUP|USE CREATOR SUBGROUP",
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.
`--description` / `description`	Optional	A user friendly description of the endpoint. If the description contains spaces, 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`	Required	The endpoint platform. Allowed values are: • `LINUX64` • `SOLARIS64` • `SOLARIS_SPARC` • `AIX` • `HP-UX` • `WINDOWS`
`--type`	Required	Type of the endpoint. Allowed values are: • `ORACLE_DB` • `ORACLE_NON_DB` • `ORACLE_ACFS` • `MYSQL_DB` • `OTHER`
`--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
`--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, execute 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`.

JSON Example

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

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

   The generated input appears as follows:

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

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" : "pfitch@example.com",
      "platform" : "LINUX64",
      "type" : "ORACLE_DB",
      "subgroup" : "USE CREATOR SUBGROUP",
      "unique" : "FALSE"
    }
  }
}
```

3. Execute 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.

3.2.3 okv admin endpoint delete Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.

JSON Example

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

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

   The generated input appears as follows:

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

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" : "sales_db_ep"
       }
     }
   }
   ```

3. Execute 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"
   }
   ```

3.2.4 okv admin endpoint download Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.
`--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 genereated input appears as follows:

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

```
      "location" : "#VALUE"
    }
  }
}
```

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

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

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

3.2.5 okv admin endpoint get-enrollment-token Command

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). This token is then used 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 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 registered endpoints, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.

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 input appears as follows:

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

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. Execute 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 showing the enrollment token appears, similar to the following:

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

3.2.6 okv admin endpoint provision Command

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
```

JSON Input File Template

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

```
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.
`--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.
`--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 input appears as follows:

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

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. Execute 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 Command
  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

3.2.7 okv admin endpoint re-enroll Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.

JSON Example

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

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

The generated input appears as follows:

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

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. Execute 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"
}
```

3.2.8 okv admin endpoint re-enroll-all Command

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

Required Authorization

System Administrator role

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 input appears as follows:

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

2. Save the generate input to a file (for example, `re-enroll-all_ep.json`).
3. Execute 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"
   }
   ```

3.2.9 okv admin endpoint update Command

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

Syntax

```
okv admin endpoint update --endpoint endpoint_name --description "description" --
email email_address --platform platform --type type  --subgroup "subgroup_value"
--unique TRUE|FALSE --name new_endpoint_name
```

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",
      "type" : "#ORACLE_DB|ORACLE_NON_DB|ORACLE_ACFS|MYSQL_DB|OTHER",
```

```
      "subgroup" : "#VALUE|NO SUBGROUP|USE CREATOR SUBGROUP",
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	Name of the endpoint that you want to update. To find existing endpoints, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.
`--description` / `description`	Optional	A user-friendly description of the endpoint. If the description contains spaces, 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: • `LINUX64` • `SOLARIS64` • `SOLARIS_SPARC` • `AIX` • `HP-UX` • `WINDOWS`
`--type` / `type`	Optional	Type of the endpoint. Allowed values are: • `ORACLE_DB` • `ORACLE_NON_DB` • `ORACLE_ACFS` • `MYSQL_DB` • `OTHER`
`--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.

Parameter/Template Parameter	Required?	Description
`--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, execute 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`.
`--name` / `name`	Optional	A new name for the endpoint. See Naming Guidelines for Objects.

JSON Example

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

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

   The generated input appears as follows:

   ```
   {
     "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",
         "type" : "#ORACLE_DB|ORACLE_NON_DB|ORACLE_ACFS|MYSQL_DB|OTHER",
   ```

```
      "subgroup" : "#VALUE|NO SUBGROUP|USE CREATOR SUBGROUP",
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

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

```
{
  "service" : {
    "category" : "admin",
    "resource" : "endpoint",
    "action" : "update",
    "options" : {
      "endpoint" : "hr_db_ep",
      "name" : "HR_DB"
    }
  }
}
```

3. Execute 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",
  "value" : {
    "status" : "PENDING",
    "locatorID" : "C27E950A-0DF3-402E-BB40-4903FC936C85"
  }
}
```

This example shows the output for renaming an endpoint in a multi-master cluster. On renaming, an endpoint is placed into the `PENDING` state for the duration of the naming conflict resolution.

Unless you renamed the endpoint in a multi-master cluster, the `status` and `locatorID` entries are not included in the output.

4
Access Management Commands

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

- okv manage-access endpoint-group add-endpoint Command
  The `okv manage-access endpoint-group add-endpoint` command adds an existing endpoint to an endpoint group.
- okv manage-access endpoint-group check-status Command
  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 Command
  The `okv manage-access endpoint-group create` command creates a new endpoint group.
- okv manage-access endpoint-group delete Command
  The `okv manage-access endpoint-group delete` command deletes an endpoint group.
- okv manage-access endpoint-group remove-endpoint Command
  The `okv manage-access endpoint-group remove-endpoint` command removes an endpoint from an endpoint group.
- okv manage-access endpoint-group update Command
  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 Command
  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 check-status Command
  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 Command
  The `okv manage-access wallet create` command creates a wallet.
- okv manage-access wallet delete Command
  The `okv manage-access wallet delete` command deletes a wallet.
- okv manage-access wallet get-default Command
  The `okv manage-access wallet get-default` command gets the default wallet that has been associated with an endpoint.
- okv manage-access wallet list-endpoint-wallets Command
  The `okv manage-access wallet list-endpoint-wallets` command lists the wallets that are associated with an endpoint.
- okv manage-access wallet remove-access Command
  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 set-default Command
  The `okv manage-access wallet set-default` command sets the default wallet for an endpoint.
- okv manage-access wallet update Command
  The `okv manage-access wallet update` command updates a wallet.
- okv manage-access wallet update-access Command
  The `okv manage-access wallet update-access` command updates the level of access that an endpoint or an endpoint group has to a wallet.

4.1 okv manage-access endpoint-group add-endpoint Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoint Groups page.
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.

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 input appears as follows:

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

2. Save the generated input to a file (for example, `add_ep_to_group.json`) and then edit it so that you can 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. Execute 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"
   }
   ```

4.2 okv manage-access endpoint-group check-status Command

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 still 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, in the Oracle Key Vault management console, select the **Endpoints** tab, then **Endpoint Groups** in the left navigation bar, and in the Endpoint Groups page, check the Endpoints Groups page. To find the locator ID in the Oracle Key Vault management console, select the **Cluster** tab and then in the left navigation bar, select **Conflict Resolution**. In the Keys, Secrets & Objects table, check the **Unique Identifier** column.

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 input appears as follows:

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

```
      "locatorID" : "#VALUE"
    }
  }
}
```

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. Execute 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.

4.3 okv manage-access endpoint-group create Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoint Groups page.
`--description` / `description`	Optional	A user-friendly description of the endpoint group 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 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, execute 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 input appears as follows:

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

```
      "description" : "#VALUE",
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

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. Execute 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`, this command also displays the object name after the conflict-name resolution.

4.4 okv manage-access endpoint-group delete Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoint Groups page.

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 input appears as follows:

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

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. Execute 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"
   }
   ```

4.5 okv manage-access endpoint-group remove-endpoint Command

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 endpoint groups, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoint Groups page.
`--endpoint` / `endpoint`	Required	Name of the endpoint that is associated with the endpoint group. To find existing endpoints, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.

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 input appears as follows:

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

2. Save the generated input to a file (for example, `remove_ep_from_epg.json`) and then edit it so that you can 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. Execute 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"
}
```

4.6 okv manage-access endpoint-group update Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoint Groups page.
`--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, execute 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 input appears as follows:

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

```
      "name" : "#VALUE",
      "description" : "#VALUE",
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

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. Execute 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.

4.7 okv manage-access wallet add-access Command

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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`--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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page. For endpoint groups, check the Endpoint Groups page.
`--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 input appears as follows. This output includes wallet access settings for both endpoints and endpoint groups. When you edit it, you must include either the endpoint settings or the endpoint group settings, but not both.

```
{
  "service": {
    "category": "manage-access",
    "resource": "wallet",
    "action": "add-access",
    "options": {
      "wallet": "#VALUE",
      "endpointGroup": "#VALUE",
      "endpoint": "#VALUE",
      "access": "#RO|RM|RO_MW|RM_MW"
    }
  }
}
```

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 or endpoint group. The following example is for the 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. Execute 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"
}
```

4.8 okv manage-access wallet check-status Command

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 still 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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar. To find the locator ID in the Oracle Key Vault management console, select the **Cluster** tab and then in the left navigation bar, select **Conflict Resolution**. In the Keys, Secrets & Objects table, check the **Unique Identifier** column.

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 input appears as follows:

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

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. Execute 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.

4.9 okv manage-access wallet create Command

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 --wallet wallet_name --description
"wallet_description" --unique TRUE|FALSE
```

JSON Input File Template

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "create",
    "options" : {
      "wallet" : "#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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar. See 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.

JSON Example

1. Generate JSON input for the `okv manage-access wallet create` command.

   ```
   okv manage-access wallet create --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "create",
       "options" : {
         "wallet" : "#VALUE",
         "description" : "#VALUE",
   ```

```
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

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. Execute 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 similar to the following:

```
{
  "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

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

4.10 okv manage-access wallet delete Command

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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.

JSON Example

1. Generate JSON input for the `okv manage-access wallet delete` command.

   ```
   okv manage-access wallet delete --generate-json-input
   ```

   The generated input appears as follows:

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

2. Save the generated input to a file (for example, `del_wallet.json`) and then edit it so that you can delete the wallet from Oracle Key Vault.

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource ": "wallet",
       "action" : "delete",
       "options" : {
         "wallet" : "hr_wallet"
       }
     }
   }
   ```

3. Execute 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"
}
```

4.11 okv manage-access wallet get-default Command

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, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "get-default",
       "options" : {
   ```

```
      "endpoint" : "#VALUE"
    }
  }
}
```

2. Save the generated input to a file (for example, `get_def_wallet.json`) and then edit it so that you can get 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. Execute 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"
  }
}
```

4.12 okv manage-access wallet list-endpoint-wallets Command

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"
```

```
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--endpoint` / `endpoint`	Required	The name of the endpoint. To find existing endpoints, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page

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 input appears as follows:

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

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. Execute 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" ]
     }
   }
   ```

4.13 okv manage-access wallet remove-access Command

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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`--endpoint` / `endpoint` or `--endpoint-group` / `endpointGroup`	Required	Name of the endpoint or endpoint group. To find existing endpoints or endpoint groups, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints or Endpoint Groups page.

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 input appears as follows. This output includes the entire output, for both the endpoint and endpoint group. When you edit it, you must include the entry for either the endpoint or the endpoint group, but not both.

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

2. Save the generated input to a file (for example, `remove_wallet_access_ep.json`) and then edit it so that you can remove wallet access from an endpoint or an endpoint group. 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. Execute 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"
}
```

4.14 okv manage-access wallet set-default Command

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"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--wallet` / `wallet`	Required	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`--endpoint` / `endpoint`	Required	Name of the endpoint. To find existing endpoints, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints page.

Example

1. Generate JSON input for the `okv manage-access wallet set-default` command.

   ```
   okv manage-access wallet set-default --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "set-default",
       "options" : {
         "wallet" : "#VALUE",
         "endpoint" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `set_def_wallet.json`) and then edit it so that you can set the default wallet for an endpoint.

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "set-default",
       "options" : {
         "wallet" : "hr_wallet",
         "endpoint" : "hr_db_ep"
   ```

```
    }
  }
}
```

3. Execute 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"
}
```

4.15 okv manage-access wallet update Command

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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`--name` / `name`	Optional	A new name for the wallet. See Naming Guidelines for Objects.

Parameter/Template Parameter	Required?	Description
`--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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "manage-access",
       "resource" : "wallet",
       "action" : "update",
       "options" : {
         "wallet" : "#VALUE",
         "name" : "#VALUE",
         "description" : "#VALUE",
   ```

```
      "unique" : "#TRUE|FALSE"
    }
  }
}
```

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

Unless you renamed the wallet in a multi-master cluster, the status and locatorID entries are not included in the output.

Related Topics

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

4.16 okv manage-access wallet update-access Command

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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`--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 or endpoint groups, in the Oracle Key Vault management console, select the **Endpoints** tab and then check the Endpoints or Endpoint Groups page.
`--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 input appears as follows. This output includes wallet access settings for both endpoints and endpoint groups. When you edit it, you must include either the endpoint settings or the endpoint group settings, but not both.

```
{
  "service" : {
    "category" : "manage-access",
    "resource" : "wallet",
    "action" : "update-access",
    "options" : {
      "wallet" : "#VALUE",
      "endpointGroup" : "#VALUE",
      "endpoint" : "#VALUE",
      "access" : "#RO|RM|RO_MW|RM_MW"
    }
  }
}
```

2. Save the generated input to a file (for example, `update_wallet_access_ep.json`) and then edit it so that you can update the wallet access to an endpoint or an endpoint group. 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. Execute 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 Command
  The `okv managed-object attribute add` command adds one or more attributes to a security object.
- okv managed-object attribute delete Command
  The `okv managed-object attribute delete` command deletes one or more attributes associated with a security object.
- okv managed-object attribute get Command
  The `okv managed-object attribute get` command retrieves an attribute or list of attributes of a security object.
- okv managed-object attribute get-all Command
  The `okv managed-object attribute get-all` command retrieves all attributes of a security object.
- okv managed-object attribute list Command
  The `okv managed-object attribute list` command retrieves the names of attributes associated with a security object.
- okv managed-object attribute modify Command
  The `okv managed-object attribute modify` command modifies attributes associated with a security object.
- okv managed-object certificate get Command
  The `okv managed-object certificate get` command retrieves a digital certificate.
- okv managed-object certificate register Command
  The `okv managed-object certificate register` command registers a certificate.
- okv managed-object certificate-request get Command
  The `okv managed-object certificate-request get` command retrieves a certificate request.
- okv managed-object certificate-request register Command
  The `okv managed-object certificate-request register` command registers a certificate request object with Oracle Key Vault.
- okv managed-object custom-attribute add Command
  The `okv managed-object custom-attribute add` command adds a custom attribute to a security object.
- okv managed-object custom-attribute delete Command
  The `okv managed-object custom-attribute delete` command deletes a custom attribute of a security object.
- okv managed-object custom-attribute modify Command
  The `okv managed-object custom-attribute modify` command modifies a custom attribute of a security object.

- okv managed-object key create Command
  The `okv managed-object key create` command creates a new symmetric key.
- okv managed-object key get Command
  The `okv managed-object key get` command retrieves an encryption key.
- okv managed-object key register Command
  The `okv managed-object key register` command registers a key.
- okv managed-object object activate Command
  The `okv managed-object object activate` command activates a security object.
- okv managed-object object destroy Command
  The `okv managed-object object destroy` command requests the server to destroy the key data for a security object.
- okv managed-object object locate Command
  The `okv managed-object object locate` command locates a security object.
- okv managed-object object query Command
  The `okv managed-object object query` command identifies supported operations and objects.
- okv managed-object object revoke Command
  The `okv managed-object object revoke` command revokes a security object.
- okv managed-object opaque get Command
  The `okv managed-object opaque get` command retrieves an object that contains opaque data.
- okv managed-object opaque register Command
  The `okv managed-object opaque register` command registers an opaque security object.
- okv managed-object private-key get Command
  The `okv managed-object private-key get` command retrieves a private key.
- okv managed-object private-key register Command
  The `okv managed-object private-key register` command registers a private key.
- okv managed-object public-key get Command
  The `okv managed-object public-key get` command retrieves a public key.
- okv managed-object public-key register Command
  The `okv managed-object public-key register` command registers a public key.
- okv managed-object secret get Command
  The `okv managed-object secret get` command retrieves the secret data from a security object of type secret.
- okv managed-object secret register Command
  The `okv managed-object secret register` command registers secret data such as passwords or random seeds.
- okv managed-object wallet add-member Command
  The `okv managed-object wallet add-member` command adds a security object to a wallet as its member.
- okv managed-object wallet delete-member Command
  The `okv managed-object wallet delete-member` command deletes the membership of the managed-object from a wallet.

- okv managed-object wallet list Command
  The `okv managed-object wallet list` command lists wallets that have their access granted to the endpoint used to connect to Oracle Key Vault.

5.1 okv managed-object attribute add Command

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, execute 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

Uses JSON syntax only: `okv managed-object attribute add --generate-json-input`

You must use the JSON syntax for this command to specify the attributes. However, you can use the `--uuid` parameter at the command line with this command. 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" : "#YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#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.

`attributes`	Required	Array of attribute names. 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, execute the `okv managed-object attribute list` command.
		Attributes that you can enter are as follows:
		• `name` includes the following:
		– `value` is the name of the value.
		– `type` is either `text` or `uri`.
		• `contactInfo`
		• `activationDate`
		• `deactivationDate`
		• `protectStopDate`
		• `processStartDate`
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.
		For date values, use the following format:
		`YYYY-MM-DD HH:mm:ss` (in UTC)
		Example showing how to use the `date` command to display the time in UTC:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`

JSON Example

1. Generate JSON input for the `okv managed-object attribute add` command and save it as `add-attrib.json`.

   ```
   okv managed-object attribute add --generate-json-input > add-attrib.json;
   more add-attrib.json
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "attribute",
       "action" : "add",
       "options" : {
   ```

```
      "uuid" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#YYYY-MM-DD HH:mm:ss"
      }
    }
  }
}
```

2. Save the generated input to a file (for example, `add_attribute.json`) and then edit it so that you can add the attributes to the security object. For example:

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

3. Execute the `okv managed-object attribute add` command using the generated JSON file.

```
okv managed-object attribute add --from-json add_attrib.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "attributes" : {
      "contactInfo" : "Added",
      "deactivationDate" : "Added",
      "name" : "Added",
      "protectStopDate" : "Added"
    }
  }
}
```

5.2 okv managed-object attribute delete Command

The `okv managed-object attribute delete` command deletes one or more attributes associated with a security object.

To find the existing attributes for the security object, execute the `okv managed-object attribute list` command.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

Uses JSON syntax only: `okv managed-object attribute delete --generate-json-input`

You must use the JSON syntax for this command to specify the attributes. However, you can use the `--uuid` parameter at the command line with this command. 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": "",
        "activationDate": "",
        "deactivationDate": "",
        "protectStopDate": "",
        "processStartDate": ""
      }
    }
  }
}
```

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.

Template Parameter	Required?	Description
`attributes`	Required	Array of attribute names. 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, execute 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.) • `activationDate` • `contactInfo` • `deactivationDate` • `protectStopDate` • `processStartDate`

JSON Example

1. Generate JSON input for the `okv managed-object attribute delete` command.

   ```
   okv managed-object attribute delete --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "attribute",
       "action": "delete",
       "options": {
         "uuid": "#VALUE",
         "attributes": {
           "name": {
             "value": "#VALUE"
           },
           "contactInfo": "",
           "activationDate": "",
           "deactivationDate": "",
           "protectStopDate": "",
           "processStartDate": ""
         }
       }
     }
   }
   ```

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. Execute 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"
       }
     }
   }
   ```

5.3 okv managed-object attribute get Command

The `okv managed-object attribute get` command retrieves an attribute or list of attributes of a security object.

To find the existing attributes for the managed object, execute the `okv managed-object attribute list` command. To retrieve the value of custom attributes, execute the `okv managed-object attribute get-all` command.

Required Authorization

The endpoint must have read permission on the object.

Syntax

Uses JSON syntax only: `okv managed-object attribute get --generate-json-input`

You must use the JSON syntax for this command to specify the attributes. However, you can use the `--uuid` parameter at the command line with this command. 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

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.
`attributes`	Required	Array of attribute names. You must use the JSON syntax to specify the attributes. You cannot specify attributes at the command line. You can retrieve the value of multiple attributes by including additional optional `ATTRIBUTE_NAME` attributes. See the example. To find the existing attributes for the managed object, execute the `okv managed-object attribute list` command. To retrieve the values of all existing attributes for the managed object, execute the `okv managed-object attribute get-all` command. See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.
`customAttributes`	Optional	Array of custom attributes. You must use the JSON syntax to specify the custom attributes. You cannot specify attributes at the command line. To find the existing attributes for the managed object, execute the `okv managed-object attribute list` command.

JSON Example

1. Generate JSON input for the `okv managed-object attribute get` command.

   ```
   okv managed-object attribute get --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "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" ]
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `get_attribute.json`) and then edit it so that you can 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"
      ],
      "customAttributes" : ["x-ApplicationTag"]
    }
  }
}
```

3. Execute 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": "pfitch@example.com",
      "cryptoUsageMask": [
        "ENCRYPT",
        "DECRYPT"
      ],
      "cryptographicAlgorithm": "AES",
      "cryptographicLength": "256",
      "name": [
        {
          "type": "text",
          "value": "PROD-HRDB-MKEY"
        }
      ],
      "objectType": "Symmetric Key",
      "state": "Active"
    },
    "customAttributes": [
      {
        "index": "1",
        "name": "x-ApplicationTag",
        "type": "Text String",
        "value": "HR-Production"
      }
    ]
```

```
    }
  }
```

5.4 okv managed-object attribute get-all Command

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "attribute",
       "action" : "get-all",
       "options" : {
         "uuid" : "#VALUE"
       }
   ```

```
  }
}
```

2. Save the generated input to a file (for example, `get-all_attribute.json`) and then edit it so that you can 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. Execute the `okv managed-object attribute get-all` command using the generated JSON file.

```
okv managed-object attribute get-all --from-json get-all_attribute.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "attributes" : {
      "activationDate" : "2020-11-21 01:00:00",
      "contactInfo" : "pfitch@example.com",
      "cryptoUsageMask" : [ "ENCRYPT", "DECRYPT" ],
      "cryptographicAlgorithm" : "AES",
      "cryptographicLength" : "256",
      "deactivationDate" : "2024-12-31 01:00:00",
      "digest" : {
       "algorithm" : "SHA-256",
       "digestValue" :
"EA31657433D91BF79660525131772D838A1128FCE6B49471726EEF5844EFA3F7",
       "keyFormatType" : "RAW"
      },
      "fresh" : "Yes",
      "initialDate" : "2020-11-21 00:57:00",
      "lastChangeDate" : "2020-11-21 20:17:19",
      "name" : [ {
        "type" : "text",
        "value" : "PROD-HRDB-MKEY"
      } ],
      "objectType" : "Symmetric Key",
      "processStartDate" : "2020-11-21 00:57:00",
      "protectStopDate" : "2024-09-30 09:00:00",
      "state" : "Active"
    },
    "customAttributes" : [ {
      "index" : "1",
      "name" : "x-ApplicationTag",
      "type" : "Text String",
      "value" : "HR-Production"
    } ]
  }
}
```

5.5 okv managed-object attribute list Command

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, execute 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 input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "attribute",
       "action": "list",
       "options": {
   ```

```
      "uuid": "#VALUE"
    }
  }
}
```

2. Save the generated input to a file (for example, `list_attribute.json`) and then edit it so that you can 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. Execute the `okv managed-object attribute list` command using the generated JSON file.

```
okv managed-object attribute list --from-json list_attribute.json
```

Output similar to the following appears:

```
{
  "result": "Success",
  "value": {
    "attributes" : [
      "activationDate" ,
      "contactInfo" ,
      "cryptoUsageMask" ,
      "cryptographicAlgorithm" ,
      "cryptographicLength" ,
      "deactivationDate" ,
      "digest" ,
      "fresh" ,
      "initialDate" ,
      "lastChangeDate" ,
      "name" ,
      "objectType" ,
      "processStartDate" ,
      "protectStopDate" ,
      "state"
    ],
    "customAttributes" : [ "x-ApplicationTag" ]
  }
}
```

5.6 okv managed-object attribute modify Command

The `okv managed-object attribute modify` command modifies attributes associated with a security object.

To find the existing attributes for the managed object, execute the `okv managed-object attribute list` command.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

Uses JSON syntax only: `okv managed-object attribute modify --generate-json-input`

You must use the JSON syntax for this command to specify the attributes. However, you can use the `--uuid` parameter at the command line with this command. 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" : "#YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#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.

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, execute 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`.
		• `activationDate`
		• `contactInfo`
		• `deactivationDate`
		• `protectStopDate`
		• `processStartDate`
		See Key Management Interoperability Protocol Specification Version 1.1 for details about these attributes.
		For date values, use the following format:
		`YYYY-MM-DD HH:mm:ss` (in UTC)
		Example showing how to use the `date` command to display the time in UTC:
		`$ date --utc "+%F %T"`
		`2021-03-15 20:31:37`

JSON Example

1. Generate JSON input for the `okv managed-object attribute modify` command.

   ```
   okv managed-object attribute modify --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "attribute",
       "action" : "modify",
       "options" : {
         "uuid" : "#VALUE",
         "attributes" : {
           "name" : {
             "value" : "#VALUE",
             "newValue" : "#VALUE",
             "newType" : "#text|uri"
           },
           "contactInfo" : "#VALUE",
   ```

```
        "activationDate" : "#YYYY-MM-DD HH:mm:ss",
        "deactivationDate" : "#YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#YYYY-MM-DD HH:mm:ss"
      }
    }
  }
}
```

2. Save the generated input to a file (for example, `modify_attribute.json`) and then edit it so that you can modify attributes that you want to change that are associated with a 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"
      }
    }
  }
}
```

3. Execute the `okv managed-object attribute modify` command using the generated JSON file.

```
okv managed-object attribute modify --from-json modify_attribute.json
```

Output similar to the following appears:

```
{
  "result": "Success",
  "value": {
    "attributes": {
      "contactInfo": "Modified",
      "deactivationDate": "Modified",
      "name": "Modified",
      "protectStopDate": "Modified"
    }
  }
}
```

5.7 okv managed-object certificate get Command

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 --uuid UUID
```

JSON Input File Template Syntax

```
{
  "service": {
    "category": "managed-object",
    "resource": "certificate",
    "action": "get",
    "options": {
      "uuid": "#VALUE"
    }
  }
}
```

Parameters

Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the certificate. To find the unique identifier for the certificate, 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 certificate get` command.

   ```
   okv managed-object certificate get --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "certificate",
       "action": "get",
       "options": {
         "uuid": "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `get_cert.json`) and then edit it so that you can retrieve the certificate.

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "certificate",
       "action": "get",
       "options": {
         "uuid": "EEED2C4F-33D7-4F9A-BF02-52DD2225A43A"
       }
   ```

```
  }
}
```

3. Execute 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-----"
  }
}
```

5.8 okv managed-object certificate register Command

The `okv managed-object certificate register` command registers a certificate.

Required Authorization

None

Syntax

```
okv managed-object certificate register --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
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "certificate",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "algorithm" : "#RSA",
      "length" : "#1024,2048,4096(RSA)",
      "mask" : #[ "ENCRYPT", "DECRYPT", "WRAP_KEY", "UNWRAP_KEY", "EXPORT",
"DERIVE_KEY", "GENERATE_CRYPTOGRAM", "VALIDATE_CRYPTOGRAM", "TRANSLATE_ENCRYPT",
"TRANSLATE_DECRYPT", "TRANSLATE_WRAP", "TRANSLATE_UNWRAP" ],
      "type" : "X_509",
      "subType" : "#USER_CERT|TRUSTPOINT",
      "privateKeyUUID" : "#VALUE",
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#VALUE",
        "deactivationDate" : "#VALUE",
```

```
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Required	File path to the certificate object.
`--type` / `type`	Required	Type of certificate. Enter the following value: `X_509`
`--sub-type` / `sub-type`	Optional	Sub-type of the certificate. Choose from the following values: • `USER_CERT` • `TRUSTPOINT`
`--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`
`--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 being uploaded. Choose from the following values: • `1024` • `2048` • `4096`
`--mask` / `mask`	Required	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `GENERATE_CRYPTOGRAM` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `UNWRAP_KEY` • `VALIDATE_CRYPTOGRAM` • `WRAP_KEY`
`--privateKeyUUID` / `privateKeyUUID`	Optional	Universally unique ID (UUID) of the private key associated with the certificate object. To find the unique identifier for the key, 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
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`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, execute 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` • `activationDate` • `deactivationDate` • `processStartDate` • `protectStopDate` 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 certificate register` command.

   ```
   okv managed-object certificate register --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "certificate",
       "action" : "register",
       "options" : {
         "object" : "#VALUE",
         "algorithm" : "#RSA",
         "length" : "#1024,2048,4096(RSA)",
         "mask" : #[ "ENCRYPT", "DECRYPT", "WRAP_KEY", "UNWRAP_KEY", "EXPORT",
   "DERIVE_KEY", "GENERATE_CRYPTOGRAM", "VALIDATE_CRYPTOGRAM", "TRANSLATE_ENCRYPT",
   "TRANSLATE_DECRYPT", "TRANSLATE_WRAP", "TRANSLATE_UNWRAP" ],
         "type" : "X_509",
         "subType" : "#USER_CERT|TRUSTPOINT",
         "privateKeyUUID" : "#VALUE",
         "wallet" : "#VALUE",
         "attributes" : {
           "name" : {
             "value" : "#VALUE",
   ```

```
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

2. Save the generated input to a file (for example, `reg_cert.json`) and then edit it so that you can register the certificate.

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "certificate",
    "action" : "register",
    "options" : {
      "object" : "./cert.pem",
      "algorithm" : "RSA",
      "length" : "2048",
      "mask" : [ "ENCRYPT"],
      "type" : "X_509",
      "subType" : "USER_CERT",
      "privateKeyUUID" : "D497994E-74CD-4F60-BF7C-52F254142705",
      "wallet" : "hr_wallet",
      "attributes" : {
        "name" : {
          "value" : "FINDB-PROD-CERT",
          "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"
      }
    }
  }
}
```

3. Execute 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"
  }
}
```

5.9 okv managed-object certificate-request get Command

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  --uuid UUID
```

JSON Input File Template Syntax

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "certificate-request",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE"
    }
  }
}
```

Parameters

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.

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "certificate-request",
       "action" : "get",
       "options" : {
         "uuid" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `get_cert_req.json`) and then edit it to specify the UUI of the certificate request.

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "certificate-request",
    "action" : "get",
    "options" : {
      "uuid" : "BC0E9004-82E0-4FFA-BFF2-29A67DDD5C64"
    }
  }
}
```

3. Execute 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-----"
  }
}
```

5.10 okv managed-object certificate-request register Command

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 --object
certificate_requeset_file_path --type certificate_requeset_type --private-
key-uuid private_key_uuid --wallet wallet_name
```

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",
```

```
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Required	File path to the certificate request object.
`--type` / `type`	Required	Type of certificate request. Choose from the following values: • `CRMF` • `PKCS10` • `PEM` • `PGP`
`--privateKeyUUID` / `privateKeyUUID`	Optional	Universally unique ID (UUID) of the private key associated with the certificate request to be registered. To find the unique identifier for the key, 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`	Optional	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.

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, execute 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` • `activationDate` • `deactivationDate` • `processStartDate` • `protectStopDate` 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 certificate-request register` command.

   ```
   okv managed-object certificate-request register --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "certificate-request",
       "action" : "register",
       "options" : {
         "type" : "#CRMF,PKCS10,PEM,PGP",
         "object" : "#VALUE",
         "privateKeyUUID" : "#VALUE",
         "wallet" : "#VALUE",
         "attributes" : {
           "name" : {
             "value" : "#VALUE",
             "type" : "#text|uri"
           },
           "contactInfo" : "#VALUE",
           "activationDate" : "#VALUE",
           "deactivationDate" : "#VALUE",
           "processStartDate" : "#VALUE",
           "protectStopDate" : "#VALUE"
         }
       }
     }
   }
   ```

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" : "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"
      }
    }
  }
}
```

3. Execute 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"
  }
}
```

5.11 okv managed-object custom-attribute add Command

The `okv managed-object custom-attribute add` command adds a custom attribute to a security object.

To find the existing attributes for the managed object, execute the `okv managed-object attribute list` command.

Required Authorization

The endpoint must have read-modify permission on the object.

Syntax

Uses JSON syntax only: `okv managed-object custom-attribute add --generate-json-input`

You must use the JSON syntax for this command to specify the attributes. However, you can use the `--uuid` parameter at the command line with this command. 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"
      }
    }
  }
}
```

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.
`customAttribute`	Required	Custom attribute name. Include the prefix `x-` in the attribute name. 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 the managed object, execute 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 either `text` or `number`. 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 add` command.

   ```
   okv managed-object custom-attribute add --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "custom-attribute",
       "action" : "add",
       "options" : {
         "uuid" : "#VALUE",
         "customAttribute" : {
           "name" : "#VALUE",
           "value" : "#VALUE",
           "type" : "#TEXT|NUMBER"
         }
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `add_cust_attr.json`) and then edit it so that you can add 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. Execute 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"
   }
   ```

5.12 okv managed-object custom-attribute delete Command

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

Uses JSON syntax only: `okv managed-object custom-attribute delete --generate-json-input`

You must use the JSON syntax for this command to specify the attributes. However, you can use the `--uuid` parameter at the command line with this command. 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.

Parameter/Template Parameter	Required?	Description
`customAttribute`	Required	Custom attribute name and its 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 a managed object, execute 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. 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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "custom-attribute",
       "action" : "delete",
       "options" : {
         "uuid" : "#VALUE",
         "customAttribute" : {
           "name" : "#VALUE",
           "index" : "#VALUE"
         }
       }
     }
   }
   ```

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. Execute 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"
}
```

5.13 okv managed-object custom-attribute modify Command

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

Uses JSON syntax only: `okv managed-object custom-attribute modify --generate-json-input`

You must use the JSON syntax for this command to specify the attributes. However, you can use the `--uuid` parameter at the command line with this command. 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"
      }
    }
  }
}
```

Parameters

Template Parameter	Required?	Description
`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.

Template Parameter	Required?	Description
`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, execute 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. 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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "custom-attribute",
       "action" : "modify",
       "options" : {
         "uuid" : "#VALUE",
         "customAttribute" : {
           "name" : "#VALUE",
           "newValue" : "#VALUE",
           "index" : "#VALUE"
         }
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `modify_cust_attr.json`) and then edit it so that you can 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. Execute 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"
   }
   ```

5.14 okv managed-object key create Command

The `okv managed-object key create` command creates a new symmetric key.

Required Authorization

None

Syntax

```
okv managed-object key create --algorithm  cryptographic_algorithm --
length key_length --mask crypographic_usage_mask --wallet wallet_name
```

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",
        "UNWRAP_KEY",
        "EXPORT",
        "DERIVE_KEY",
        "GENERATE_CRYPTOGRAM",
        "VALIDATE_CRYPTOGRAM",
        "TRANSLATE_ENCRYPT",
        "TRANSLATE_DECRYPT",
        "TRANSLATE_WRAP",
        "TRANSLATE_UNWRAP"
      ],
      "wallet": "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--algorithm` / `algorithm`	Required	Cryptographic algorithm. Choose from the following values: • `AES` • `3DES`
`--length` / `length`	Required	Key length for the algorithm. Choose from the following values: • For `AES`: `128`, `192`, `256` • For `3DES`: `112`, `168`
`--mask` / `mask`	Required	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `GENERATE_CRYPTOGRAM` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `UNWRAP_KEY` • `VALIDATE_CRYPTOGRAM` • `WRAP_KEY`
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.

JSON Example

1. Generate JSON input for the `okv managed-object key create` command.

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

   The generated input appears 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",
           "WRAP_KEY",
           "UNWRAP_KEY",
   ```

```
            "EXPORT",
            "DERIVE_KEY",
            "GENERATE_CRYPTOGRAM",
            "VALIDATE_CRYPTOGRAM",
            "TRANSLATE_ENCRYPT",
            "TRANSLATE_DECRYPT",
            "TRANSLATE_WRAP",
            "TRANSLATE_UNWRAP"
          ],
          "wallet": "#VALUE"
        }
      }
}
```

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

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

3. Execute 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"
  }
}
```

5.15 okv managed-object key get Command

The `okv managed-object key get` command retrieves an encryption key.

Required Authorization

The endpoint must have read permission on the key object.

Syntax

```
okv managed-object key get --uuid UUID
```

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "key",
    "action": "get",
    "options": {
      "uuid": "#VALUE"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the key. To find the unique identifier for the key, 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 key get` command.

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

   The generated input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "key",
       "action": "get",
       "options": {
         "uuid": "#VALUE"
       }
     }
   }
   ```

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

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "key",
       "action": "get",
       "options": {
         "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
       }
     }
   }
   ```

3. Execute 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"
  }
}
```

5.16 okv managed-object key register Command

The `okv managed-object key register` command registers a key.

Required Authorization

None

Syntax

```
okv managed-object key register --algorithm cryptographic_algorithm --
length key_length --mask cryptographic_usage_mask --object key_file_path --
wallet wallet_name
```

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",
      "mask" : #[ "ENCRYPT", "DECRYPT", "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" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--algorithm` / `algorithm`	Required	Cryptographic algorithm. Choose from the following values: • `AES` • `3DES`
`--length` / `length`	Required	Key length for the algorithm. Choose from the following values: • For `AES`: `128`, `192`, `256` • For `3DES`: `112`, `168`
`--mask` / `mask`	Required	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `GENERATE_CRYPTOGRAM` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `UNWRAP_KEY` • `VALIDATE_CRYPTOGRAM` • `WRAP_KEY`
`--object` / `object`	Required	File path to the symmetric key object.
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.

Parameter/Template Parameter	Required?	Description
attributes	Required	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, execute 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` • `activationDate` • `deactivationDate` • `processStartDate` • `protectStopDate` 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 key register` command.

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

The generated input appears as follows:

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "key",
    "action" : "register",
    "options" : {
      "length" : "#112,168(3DES)|128,192,256(AES)",
      "object" : "#VALUE",
      "algorithm" : "#3DES|AES",
      "mask" : #[ "ENCRYPT", "DECRYPT", "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" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
```

```
    }
  }
}
```

2. Save the generated input to a file (for example, `reg_key.json`) and then edit it so that you can 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"
      }
    }
  }
}
```

3. Execute 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"
  }
}
```

5.17 okv managed-object object activate Command

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.

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 input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "object",
       "action": "activate",
       "options": {
         "uuid": "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `activate_object.json`) and then edit it so that you can activate the security object.

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

```
        "uuid": "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
      }
    }
}
```

3. Execute 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"
   }
   ```

5.18 okv managed-object object destroy Command

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.

JSON Example

1. Generate JSON input for the `okv managed-object object destroy` command.

   ```
   okv managed-object object destroy --generate-json-input
   ```

The generated input appears as follows:

```
{
  "service": {
    "category": "managed-object",
    "resource": "object",
    "action": "destroy",
    "options": {
      "uuid": "#VALUE"
    }
  }
}
```

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. Execute the `okv managed-object object destroy` command using the generated JSON file.

```
okv managed-object object destroy --from-json del_obj.json
```

Output similar to the following appears:

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

5.19 okv managed-object object locate Command

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 --max max_value  --object-group-member
object_group_member_type --state state_value --name name_value
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "object",
    "action" : "locate",
    "options" : {
      "max" : "#VALUE",
      "objectGroupMember" : "#FRESH|DEFAULT",
```

```
      "attributes" : {
        "name" : {
          "value" : "#VALUE"
        },
        "state" :
"#PREACTIVE|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",
        "deactivationDate" : "#YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#YYYY-MM-DD HH:mm:ss",
        "initialDate" : "#YYYY-MM-DD HH:mm:ss",
        "lastChangeDate" : "#YYYY-MM-DD HH:mm:ss",
        "compromiseDate" : "#YYYY-MM-DD HH:mm:ss",
        "compromiseOccurrenceDate" : "#YYYY-MM-DD HH:mm:ss",
        "destroyDate" : "#YYYY-MM-DD HH:mm:ss",
        "archiveDate" : "#YYYY-MM-DD HH:mm:ss"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER"
      } ]
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--max` / `max`	Required	Maximum number of objects that this command should return
`--object-group-member` / `object-group-member`	Optional	Enter one of the following group values: • `DEFAULT` • `FRESH`

Parameter/Template Parameter	Required?	Description
`--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` / `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. • `activationDate` activation date of the object. • `deactivationDate` is the deactivateion date of the object. • `processStartDate` is the process start date of the object. • `protectStopDate` is the protect stop date of the object. • `initialDate` is the initial date of the object. • `lastChangeDate` is the last change date of the object. • `compromiseDate` is the compromise date of the object.

Parameter/Template Parameter	Required?	Description
		• `compromiseOccurrenceDate` is the compromise occurrence date of the object. • `destroyDate` is the destroy date of the object. • `archiveDate` is the archive date of the object.
`customAttributes`	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 either `text` or `number`. 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 object locate` command.

```
okv managed-object object locate --generate-json-input
```

The generated input appears as follows:

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "object",
    "action" : "locate",
    "options" : {
      "max" : "#VALUE",
      "objectGroupMember" : "#FRESH|DEFAULT",
      "attributes" : {
        "name" : {
          "value" : "#VALUE"
        },
        "state" :
"#PREACTIVE|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",
        "deactivationDate" : "#YYYY-MM-DD HH:mm:ss",
        "processStartDate" : "#YYYY-MM-DD HH:mm:ss",
        "protectStopDate" : "#YYYY-MM-DD HH:mm:ss",
        "initialDate" : "#YYYY-MM-DD HH:mm:ss",
        "lastChangeDate" : "#YYYY-MM-DD HH:mm:ss",
        "compromiseDate" : "#YYYY-MM-DD HH:mm:ss",
        "compromiseOccurrenceDate" : "#YYYY-MM-DD HH:mm:ss",
        "destroyDate" : "#YYYY-MM-DD HH:mm:ss",
        "archiveDate" : "#YYYY-MM-DD HH:mm:ss"
      },
      "customAttributes" : [ {
        "name" : "#VALUE",
        "value" : "#VALUE",
        "type" : "#TEXT|NUMBER"
      } ]
    }
  }
}
```

2. Save the generated input to a file (for example, `locate-obj.json`) and then edit it so that you can 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"
           }
      },
     "customAttributes" : [{
        "name": "x-test_1",
        "value": "test_1",
        "type": "TEXT"
       },
       {
        "name": "x-number",
        "value": "1",
        "type": "NUMBER"
       }]
    }
  }
}
```

3. Execute 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" ]
  }
}
```

5.20 okv managed-object object query Command

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 input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "object",
       "action": "query"
     }
   }
   ```

2. Save the generated input to a file (for example, `query-obj.json`).
3. Execute 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"
    ],
    "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"
    ]
  }
}
```

5.21 okv managed-object object revoke Command

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 --compromiseDate 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`	Required	Enter one of the following values: • `AFFILIATION_CHANGED` • `CA_COMPROMISE` (Certificate authority compromise) • `CESSATION_OF_OPERATION` • `KEY_COMPROMISE` • `PRIVILEGE_WITHDRAWN` • `SUPERSEDED` • `UNSPECIFIED`
`--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.

JSON Example

1. Generate JSON input for the `okv managed-object object revoke` command.

   ```
   okv managed-object object revoke --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "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"
       }
     }
   }
   ```

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. Execute 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"
   }
   ```

5.22 okv managed-object opaque get Command

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 --uuid UUID
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "opaque",
    "action" : "get",
    "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 opaque get` command.

   ```
   okv managed-object opaque get --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "opaque",
       "action" : "get",
       "options" : {
         "uuid" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `get_opaque_object.json`) and then edit it so that you can retrieve the data from the opaque object.

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "opaque",
       "action" : "get",
       "options" : {
         "uuid" : "2359E04F-DA61-4F7C-BF9F-913D3369A93A"
       }
     }
   }
   ```

3. Execute 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" :
   "2D2D2D2D2D424547494E20505249564154452044B45592D2D2D2D2D0A4D494945765149424144
   414E42676B71686B6947397730424151454641415343424B637767675363
   <<<< Output Truncated>>>>
   706753317063363465 6D3630686C72336B786C593858665734317A594A450A724546334C652F4
   ```

```
A4F4B4968674A754C367352734C67553D0A2D2D2D2D2D454E442050524956415445204B45592D2D2D2D
2D0A"
  }
}
```

5.23 okv managed-object opaque register Command

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 --object` *`object_name`* `--wallet` *`wallet_name`*

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "opaque",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Required	File path to the object. To find the names of existing objects to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Keys, Secrets & Objects** in the left navigation bar.
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.

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, execute 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` • `activationDate` • `deactivationDate` • `processStartDate` • `protectStopDate` 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 opaque register` command.

   ```
   okv managed-object opaque register --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "opaque",
       "action" : "register",
       "options" : {
         "object" : "#VALUE",
         "wallet" : "#VALUE",
         "attributes" : {
           "name" : {
             "value" : "#VALUE",
             "type" : "#text|uri"
           },
           "contactInfo" : "#VALUE",
           "activationDate" : "#VALUE",
           "deactivationDate" : "#VALUE",
           "processStartDate" : "#VALUE",
           "protectStopDate" : "#VALUE"
         }
       }
     }
   }
   ```

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

3. Execute 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"
   }
   ```

Related Topics

- okv managed-object opaque get Command
  The `okv managed-object opaque get` command retrieves an object that contains opaque data.

5.24 okv managed-object private-key get Command

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  --uuid UUID
```

JSON Input File Template Syntax

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "private-key",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE"
```

```
    }
  }
}
```

Parameters

Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the private key. To find the unique identifier for the key, 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 private-key get` command.

   ```
   okv managed-object private-key get  --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "private-key",
       "action" : "get",
       "options" : {
         "uuid" : "#VALUE"
       }
     }
   }
   ```

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. Execute 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-----"
  }
}
```

5.25 okv managed-object private-key register Command

The `okv managed-object private-key register` command registers a private key.

Required Authorization

None

Syntax

```
okv managed-object private-key register --object private_key_file_path --
algorithm cryptographic_algorithm --length key_length --mask
cryptographic_usage_mask --wallet wallet_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "private-key",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "algorithm" : "#RSA",
      "length" : "#1024,2048,4096(RSA)",
      "mask" : #[ "ENCRYPT", "DECRYPT", "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" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Required	File path to the private key object.
`--algorithm` / `algorithm`	Required	Cryptographic algorithm. The following value is valid: `RSA`

Parameter/Template Parameter	Required?	Description
`--length` / `length`	Required	Key length for the algorithm. Choose from the following values: • `1024` • `2048` • `4096`
`--mask` / `mask`	Required	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `GENERATE_CRYPTOGRAM` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `UNWRAP_KEY` • `VALIDATE_CRYPTOGRAM` • `WRAP_KEY`
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`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, execute 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` • `activationDate` • `deactivationDate` • `processStartDate` • `protectStopDate` 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 private-key register` command.

```
okv managed-object private-key register --generate-json-input
```

The generated input appears as follows:

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "private-key",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "algorithm" : "#RSA",
      "length" : "#1024,2048,4096(RSA)",
      "mask" : #[ "ENCRYPT", "DECRYPT", "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" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

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" : "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"
      }
    }
  }
}
```

3. Execute 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"
     }
   }
   ```

5.26 okv managed-object public-key get Command

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  --uuid UUID
```

JSON Input File Template Syntax

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "public-key",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE"
    }
  }
}
```

Parameters

Template Parameter	Required?	Description
`--uuid` / `uuid`	Required	Universally unique ID (UUID) of the public key. To find the unique identifier for the key, 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 public-key get` command.

   ```
   okv managed-object public-key get --generate-json-input
   ```

   The generated input appears as follows:

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "public-key",
    "action" : "get",
    "options" : {
      "uuid" : "#VALUE"
    }
  }
}
```

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. Execute 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-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtK4YrT6A/
4tVnadRg0ZT\nprsdUwXrIdoqf1+ye/
yVkN6RmtR7mthn6WIIrbTVX5MuAkLc6yyuMEc+nLDPZzrU\nFXkCAQeVR7sT/
hQo74dQHebIfJxgx+uZrlzOgT4Il1qfmjR6y81RjTvAU8ZPdzPb\nuXKHZErZVQdoXUw5uFrTNzOegLbYJF
I2dZnf3erB7Ho64DckFRoFP05cc3A0iLrL\ntzE8CcjAlBlXTGJD4kAtTEet/
0TkvuHzBHr23zkfj0kWV3PHGYYC3O+/UzXg/nal\n3iTK5yRDkln45AyI/PkfzAFiZ/
kX9C66H0WRMxgfaOn/uRNbikFOFK6IPOGcT+0S\n/QIDAQAB\n-----END PUBLIC KEY-----"
  }
}
```

5.27 okv managed-object public-key register Command

The `okv managed-object public-key register` command registers a public key.

Required Authorization

None

Syntax

```
okv managed-object public-key register --object public_key_file_path --algorithm
cryptographic_algorithm --length key_length --mask cryptographic_usage_mask --
private-key-uuid private_key_uuid --wallet wallet_name
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "public-key",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "algorithm" : "#RSA",
      "length" : "#1024,2048,4096(RSA)",
      "mask" : #[ "ENCRYPT", "DECRYPT", "WRAP_KEY", "UNWRAP_KEY", "EXPORT",
"DERIVE_KEY", "GENERATE_CRYPTOGRAM", "VALIDATE_CRYPTOGRAM", "TRANSLATE_ENCRYPT",
"TRANSLATE_DECRYPT", "TRANSLATE_WRAP", "TRANSLATE_UNWRAP" ],
      "privateKeyUUID" : "#VALUE",
      "wallet" : "#VALUE",
      "attributes" : {
        "name" : {
          "value" : "#VALUE",
          "type" : "#text|uri"
        },
        "contactInfo" : "#VALUE",
        "activationDate" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--object` / `object`	Required	File path to the public key object.
`--algorithm` / `algorithm`	Required	Cryptographic algorithm. The following value is valid: • `RSA`
`--length` / `length`	Required	Key length for the algorithm. Choose from the following values: • `1024` • `2048` • `4096`

Parameter/Template Parameter	Required?	Description
`--mask` / `mask`	Required	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `GENERATE_CRYPTOGRAM` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `UNWRAP_KEY` • `VALIDATE_CRYPTOGRAM` • `WRAP_KEY`
`--privateKeyUUID` / `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, 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`	Optional	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`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, execute 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` • `activationDate` • `deactivationDate` • `processStartDate` • `protectStopDate` 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 public-key register` command.

   ```
   okv managed-object public-key register --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "managed-object",
       "resource" : "public-key",
       "action" : "register",
       "options" : {
         "object" : "#VALUE",
         "algorithm" : "#RSA",
         "length" : "#1024,2048,4096(RSA)",
         "mask" : #[ "ENCRYPT", "DECRYPT", "WRAP_KEY", "UNWRAP_KEY", "EXPORT",
   "DERIVE_KEY", "GENERATE_CRYPTOGRAM", "VALIDATE_CRYPTOGRAM",
   "TRANSLATE_ENCRYPT", "TRANSLATE_DECRYPT", "TRANSLATE_WRAP",
   "TRANSLATE_UNWRAP" ],
         "privateKeyUUID" : "#VALUE",
         "wallet" : "#VALUE",
         "attributes" : {
           "name" : {
             "value" : "#VALUE",
             "type" : "#text|uri"
           },
           "contactInfo" : "#VALUE",
           "activationDate" : "#VALUE",
           "deactivationDate" : "#VALUE",
           "processStartDate" : "#VALUE",
           "protectStopDate" : "#VALUE"
         }
       }
     }
   }
   ```

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" : "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"
      }
    }
  }
}
```

3. Execute 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 "
  }
}
```

5.28 okv managed-object secret get Command

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 --uuid UUID
```

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "secret",
    "action": "get",
    "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 secret get` command.

   ```
   okv managed-object secret get --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "secret",
       "action": "get",
       "options": {
         "uuid": "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `secret_get.json`) and then edit it so that you can locate the secret object.

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "secret",
       "action": "get",
       "options": {
         "uuid": "D69D2F32-2DBB-4FF3-BF52-95487526E6EC"
       }
     }
   }
   ```

3. Execute 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"
     }
   }
   ```

5.29 okv managed-object secret register Command

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 --object object_name type PASSWORD|SEED
wallet wallet_name --mask cryptogrpahic_usage_mask
```

JSON Input File Template

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "secret",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "type" : "#PASSWORD|SEED",
      "mask" : #[ "ENCRYPT", "DECRYPT", "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" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
    }
  }
}
```

Parameters

Parameter/Template	Required?	Description
`--object` / `object`	Required	Path of the object file containing secret data.
`--type` / `type`	Required	Enter one of the following values: • `PASSWORD` • `SEED`
`--wallet` / `wallet`	Optional	Wallet name. To find the names of existing wallets to which you have access, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.
`--mask` / `mask`	Required	Cryptographic usage mask, enclosed in double quotation marks. Choose from the following values: • `ENCRYPT` • `DECRYPT` • `DERIVE_KEY` • `EXPORT` • `GENERATE_CRYPTOGRAM` • `TRANSLATE_DECRYPT` • `TRANSLATE_ENCRYPT` • `TRANSLATE_UNWRAP` • `TRANSLATE_WRAP` • `UNWRAP_KEY` • `VALIDATE_CRYPTOGRAM` • `WRAP_KEY`

Parameter/Template	Required?	Description
`attributes`	Required	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, execute 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` • `activationDate` • `deactivationDate` • `processStartDate` • `protectStopDate` 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 secret register` command.

```
okv managed-object secret register --generate-json-input
```

The generated input appears as follows:

```
{
  "service" : {
    "category" : "managed-object",
    "resource" : "secret",
    "action" : "register",
    "options" : {
      "object" : "#VALUE",
      "type" : "#PASSWORD|SEED",
      "mask" : #[ "ENCRYPT", "DECRYPT", "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" : "#VALUE",
        "deactivationDate" : "#VALUE",
        "processStartDate" : "#VALUE",
        "protectStopDate" : "#VALUE"
      }
```

```
    }
  }
}
```

2. Save the generated input to a file (for example, `reg-secret.json`) and then edit it so that you can 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" : "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"
      }
    }
  }
}
```

3. Execute 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"
  }
}
```

5.30 okv managed-object wallet add-member Command

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 --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 managed 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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation bar.

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 input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "wallet",
       "action": "add-member",
       "options": {
         "uuid": "#VALUE",
         "wallet": "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `add_wallet_member.json`) and then edit it so that you can add a security object to a wallet.

   ```
   {
     "service": {
       "category": "managed-object",
   ```

```
    "resource": "wallet",
    "action": "add-member",
    "options": {
      "uuid": "D69D2F32-2DBB-4FF3-BF52-95487526E6EC",
      "wallet": "hr_wallet"
    }
  }
}
```

3. Execute 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"
}
```

5.31 okv managed-object wallet delete-member Command

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 --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 managed 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, in the Oracle Key Vault management console, select the **Keys & Wallets** tab, and then click **Wallets** in the left navigation panel.

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 input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "wallet",
       "action": "delete-member",
       "options": {
         "uuid": "#VALUE",
         "wallet": "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `delete_wallet_member.json`) and then edit it so that you can delete a security object from a wallet.

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "wallet",
       "action": "delete-member",
       "options": {
         "uuid": "D69D2F32-2DBB-4FF3-BF52-95487526E6EC",
         "wallet": "hr_wallet"
       }
     }
   }
   ```

3. Execute 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"
}
```

5.32 okv managed-object wallet list Command

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

JSON Input File Template

```
{
  "service": {
    "category": "managed-object",
    "resource": "wallet",
    "action": "list"
  }
}
```

```
okv managed-object wallet list
```

Parameters

None

JSON Example

1. Generate JSON input for the `okv managed-object wallet list` command.

   ```
   okv managed-object wallet list --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service": {
       "category": "managed-object",
       "resource": "wallet",
       "action": "list"
     }
   }
   ```

2. Save the generated input to a file (for example, `wallet_list.json`).
3. Execute 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"
    ]
  }
}
```

6
Monitoring Commands

You can use the monitoring commands to check the Oracle Key Vault configuration, health, and deployment modes.

- okv cluster info get Command
  The `okv cluster info get` command retrieves status information about a cluster or a cluster node.
- okv cluster status get Command
  The `okv cluster status get` command retrieves dynamic information about the cluster or the specified cluster node.
- okv primary-standby info get Command
  The `okv primary-standby info get` command displays static information about the Oracle Key Vault primary-standby configuration.
- okv primary-standby status get Command
  The `okv primary-standby status get` command retrieves dynamic information about the Oracle Key Vault primary-standby configuration.
- okv server info get Command
  The `okv server info get` command retrieves static information about an Oracle Key Vault server.
- okv server status get Command
  The `okv server status get` command retrieves status information about an Oracle Key Vault server.

6.1 okv cluster info get Command

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

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "cluster",
       "resource" : "info",
       "action" : "get",
       "options" : {
         "node" : "#VALUE"
       }
     }
   }
   ```

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. Execute 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" : "subgrp1",
    "disableDate" : "",
    "ipAddress" : "192.0.2.114",
    "joinDate" : "2021-06-16 06:57:45",
    "mode" : "Read-Write",
    "nodeID" : "1",
    "nodeName" : "node1",
    "readWritePeer" : "node2",
    "status" : "ACTIVE",
    "version" : "21.2.0.0.0"
  }
}
```

- **Get cluster information for all nodes in the cluster:**

```
{
  "result" : "Success",
  "value" : {
    "clusterName" : "cluster1",
    "clusterSubgroups" : [ "subgrp1", "subgrp2" ],
    "clusterVersion" : "21.2.0.0.0",
    "maximumDisableNodeDuration" : "24 hrs",
    "nodes" : [ {
      "nodeName" : "node1",
      "nodeID" : "1",
      "ipAddress" : "192.0.2.114",
      "mode" : "Read-Write",
      "status" : "ACTIVE",
      "readWritePeer" : "node2",
      "clusterSubgroup" : "subgrp1",
      "joinDate" : "2021-06-16 06:57:45",
      "disableDate" : "",
      "version" : "21.2.0.0.0"
    }, {
      "nodeName" : "node2",
      "nodeID" : "2",
      "ipAddress" : "192.0.2.115",
      "mode" : "Read-Write",
      "status" : "ACTIVE",
      "readWritePeer" : "node1",
      "clusterSubgroup" : "subgrp2",
      "joinDate" : "2021-06-16 07:05:34",
      "disableDate" : "",
      "version" : "21.2.0.0.0"
```

```
      } ]
    }
  }
```

6.2 okv cluster status get Command

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

System Administrator role

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "cluster",
       "resource" : "status",
   ```

```
    "action" : "get",
    "options" : {
      "node" : "#VALUE"
    }
  }
}
```

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. Execute 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" : "Could not determine",
         "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" : "Could not determine"
    }, {
      "nodeID" : "2",
      "nodeName" : "node2",
      "mode" : "Read-Write",
      "status" : "ACTIVE",
      "nameResolutionTime" : "119.98 sec(s)"
    } ],
    "unresolvedConflicts" : {
      "endpointNameConflicts" : "0",
      "endpointGroupNameConflicts" : "0",
      "userNameConflicts" : "0",
      "userGroupNameConflicts" : "0",
      "walletNameConflicts" : "0",
      "kmipObjectNameConflicts" : "0"
    }
  }
}
```

6.3 okv primary-standby info get Command

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

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 input appears as follows:

```
{
  "service" : {
    "category" : "primary-standby",
    "resource" : "info",
    "action" : "get"
  }
}
```

2. Save the output to a file (for example, `primary-standby-info.json`).
3. Execute 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"
  }
}
```

6.4 okv primary-standby status get Command

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

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "primary-standby",
       "resource" : "status",
       "action" : "get"
     }
   }
   ```

2. Save the generated input to a file (for example, `primary-standby-status.json`).
3. Execute 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"
     }
   }
   ```

6.5 okv server info get Command

The `okv server info get` command retrieves static information about an Oracle Key Vault server.

Required Authorization

System Administrator role

`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

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "server",
       "resource" : "info",
       "action" : "get"
     }
   }
   ```

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. Execute 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:

   ```
   {
     "result" : "Success",
     "value" : {
       "deploymentType" : "Cluster",
       "serverCertificateExpirationDate" : "2023-04-17 07:20:54",
       "serverTime" : "2021-04-22 05:59:38",
       "version" : "21.2.0.0.0"
     }
   }
   ```

   All dates are shown in UTC.

6.6 okv server status get Command

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

Required Authorization

System Administrator role

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "server",
       "resource" : "status",
       "action" : "get"
     }
   }
   ```

2. Save the generated input to a file (for example, `server-status.json`).
3. Execute 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" : "No successful backup done",
       "freeSpace" : "88%",
       "uptime" : "00:55 HH:MM"
     }
   }
   ```

7

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 Command
  The `okv backup destination create` command creates a remote backup destination for the Oracle Key Vault server.
- okv backup destination delete Command
  The `okv backup destination delete` command deletes a backup destination.
- okv backup destination get Command
  The `okv backup destination get` command gets information about a specific backup destination.
- okv backup destination list Command
  The `okv backup destination list` command displays a list of the names of the current Oracle Key Vault server backup destinations.
- okv backup destination list-backups Command
  The `okv backup destination list-backups` command lists the backups that are available for restore operations on a destination.
- okv backup destination update Command
  The `okv backup destination update` command updates saved backup destination settings.
- okv backup destination get-public-key Command
  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 reset-host-key Command
  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 history list Command
  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 schedule create Command
  The `okv backup schedule create` command creates a backup schedule job.
- okv backup schedule delete Command
  The `okv backup schedule delete` command deletes scheduled backup job.
- okv backup schedule get Command
  The `okv backup schedule get` command retrieves detailed information about a scheduled Oracle Key Vault server backup.
- okv backup schedule list Command
  The `okv backup schedule list` command displays a listing of the currently scheduled Oracle Key Vault server backups.

- okv backup schedule pause Command
  The `okv backup schedule pause` command pauses a scheduled Oracle Key Vault server backup.
- okv backup schedule resume Command
  The `okv backup schedule resume` command resumes a paused Oracle Key Vault backup job.
- okv backup schedule update Command
  The `okv backup schedule update` command updates a currently scheduled backup.
- okv backup restore start Command
  The `okv backup restore start` command starts the restore process of an Oracle Key Vault backup.
- okv backup restore status Command
  The `okv backup restore status` command checks the status of the Oracle Key Vault backup restore operation.

7.1 okv backup destination create Command

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
--user-name user_name
--authentiation-method password|key-based
```

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"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup destination. For a listing of existing names, execute the `okv backup destination list` command.
`--transfer-method` / `transferMethod`	Required	Enter one of the following values: • `scp` • `sftp`
`--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.
`--port` / `port`	Required	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.
`--user-name` / `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`	Required	Enter one of the following values: • `password`: The password of the user account specified in the `--user-name` / `userName`parameter. • `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.

JSON Example

1. Generate JSON input for the `okv backup destination create` command.

   ```
   okv backup destination create --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "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"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-srvr-dest-create.json`) and then edit it so that you can create the backup server destination.

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "create",
       "options" : {
         "name" : "hr_backup",
         "transferMethod" : "scp",
         "hostName" : "192.0.2.34",
         "port" : "22",
         "path" : "/opt/okv/bckups",
         "userName" : "psmith",
         "authenticationMethod" : "password"
       }
     }
   }
   ```

3. Execute 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"
   }
   ```

7.2 okv backup destination delete Command

The `okv backup destination delete` command deletes a 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, execute 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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "delete",
       "options" : {
         "name" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-srvr-dest-del.json`) and then edit it so that you can delete the backup server destination.

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
   ```

```
    "action" : "delete",
    "options" : {
      "name" : "prod_okv_backup_dest"
    }
  }
}
```

3. Execute 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"
   }
   ```

7.3 okv backup destination get Command

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, execute 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 input appears as follows:

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "get",
    "options" : {
      "name" : "#VALUE"
    }
  }
}
```

2. Save the generated input to a file (for example, `bkup-srvr-dest-get.json`) and then edit it so that you can get the backup server destination.

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "get",
    "options" : {
      "name" : "PROD_OKV_BACKUP_DEST"
    }
  }
}
```

3. Execute 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" : "Key-based",
    "hostName" : "192.0.2.1",
    "name" : "PROD_OKV_BACKUP_DEST",
    "path" : "/net/okv_backup_dest",
    "transferMethod" : "scp",
    "userName" : "psmith"
  }
}
```

7.4 okv backup destination list Command

The `okv backup destination list` command displays a list of the names 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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "list"
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-srvr-dest.json`).

3. Execute the `okv backup destination list` command using the generated JSON file.

   ```
   okv backup destination list --from-json bkup-srvr-dest.json
   ```

   Output similar to the following appears:

   ```
   {
     "result" : "Success",
     "destinations" : [
       {
         "authenticationMethod" : " ",
         "path" : "-",
         "hostName" : "localhost",
         "name" : "LOCAL",
         "transferMethod" : "-",
         "userName" : "-"
       },
       {
         "authenticationMethod" : "Password",
         "path" : "/net/okv_backup_dest",
         "hostName" : "192.0.2.169",
         "name" : "PROD_OKV_BACKUP_DEST",
         "port" : "22",
         "transferMethod" : "scp",
         "userName" : "psmith"
       }
     ]
   }
   ```

7.5 okv backup destination list-backups Command

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, execute 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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "list-backups",
       "options" : {
         "name" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `dest-list-backups.json`) and then edit it so that you can list the backup server destinations.

   ```
   {
     "service" : {
   ```

```
    "category" : "backup",
    "resource" : "destination",
    "action" : "list-backups",
    "options" : {
      "name" : "prod_okv_backup_dest"
    }
  }
}
```

3. Execute 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_onetime_onetime_20210118175804",
         "time" : "2021-03-31 18:36:00",
         "type" : "One-Time"
       }
     ]
   }
   ```

7.6 okv backup destination update Command

The `okv backup destination update` command updates saved backup destination settings.

Required Authorization

System Administrator role

Syntax

```
okv backup destination update
--name destination_name
--port port
--user-name user_name
--authentication-method authentication_method
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "update",
    "options" : {
      "name" : "#VALUE",
      "port" : "#VALUE",
      "userName" : "#VALUE",
      "authenticationMethod" : "#password|key-based"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--name` / `name`	Required	Name of the backup destination. For a listing of existing backup destination names, execute 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.
`--authentication-method` / `authenticationMethod`	Optional	Enter one of the following values: • `password`: The password of the user account specified in the `--user-name` / `userName`parameter. • `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.

Example

1. Generate JSON input for the `okv backup destination update` command.

   ```
   okv backup destination update --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "update",
       "options" : {
         "name" : "#VALUE",
         "port" : "#VALUE",
         "userName" : "#VALUE",
         "authenticationMethod" : "#password|key-based"
       }
   ```

```
  }
}
```

2. Save the generated input to a file (for example, `update-bkup-srvr-dest.json`) and then edit it so that you can update the backup server destination.

```
{
  "service" : {
    "category" : "backup",
    "resource" : "destination",
    "action" : "update",
    "options" : {
      "name" : "hr_backup",
      "port" : "22",
      "userName" : "psmith",
      "authenticationMethod" : "password"
    }
  }
}
```

3. Execute 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"
}
```

7.7 okv backup destination get-public-key Command

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`

   ```
   okv backup destination get-public-key --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "get-public-key"
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-dest-get-public-key.json`).

3. Execute 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
   AAAAB3NzaC1yc2EAAAADAQABAAABAQDRLparTgrf5F2IExZ1IMZecobMw2ptj5WWlr6l2ww9GHZ5YgMiTNB
   CiAjr68KgLgJ9eRkOSSz7tsnYzcw8th45abB344LzMBLREqtqbV4U0PYHMmt1ovhd+djhsYnJbXptfiSAfe
   2f+1/XPlIYcZNo3m5imffgaIsrn9WlIYxOnP7rrZW3mQkPRLADElTAMWl7zDj71mZrenNBInTd70CBX/
   L7C4NABikPu1E7TxpASQRW9y/n5zdGR4TVww06nAEseCfwfzV1ToNK7CFwFWv/OdIARVVSqwCkCDrwP/
   pNYr7WjzXR939xBfuXaWNZpoDkN1Yxb5sk1NEYRT+cs/SD\n"
     }
   }
   ```

7.8 okv backup destination reset-host-key Command

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, execute 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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "reset-host-key",
       "options" : {
         "name" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `reset-host-key.json`) and then edit it so that you can reset the host key for the destination server.

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "destination",
       "action" : "reset-host-key",
       "options" : {
         "name" : "hr_backup"
       }
     }
   }
   ```

3. Execute 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"
   }
   ```

7.9 okv backup history list Command

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"
    }
  }
}
```

Parameters

Parameter/Template Parameter	Required?	Description
`--max` / `max`	Required	The maximum number of the most recent complete backup information records that should be returned

JSON Example

1. Generate JSON input for the `okv backup history list` command.

   ```
   okv backup history list --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "history",
       "action" : "list",
       "options" : {
         "max" : "#VALUE"
   ```

```
    }
  }
}
```

2. Save the generated input to a file (for example, `bkup-history-list.json`) and then edit it so that you can generate a backup history list.

```
{
  "service" : {
    "category" : "backup",
    "resource" : "history",
    "action" : "list",
    "options" : {
      "max" : "10"
    }
  }
}
```

3. Execute 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"
      }
    ]
  }
```

7.10 okv backup schedule create Command

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_in_UTC"
--destination LOCAL|VALUE
--type ONE-TIME|PERIODIC
--interval timing
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "create",
    "options" : {
      "name" : "#VALUE",
      "startTime" : "#NOW|YYYY-MM-DD HH:mm:ss",
      "type" : "#ONE-TIME|PERIODIC",
      "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, execute the `okv backup schedule list` command.

Parameter/Template Parameter	Required?	Description
`--start-time` / `startTime`	Required	Time to begin the scheduled backup, in double quotation marks. Enter one of the following values, in UTC: • `NOW` • A specific time, using the following format: `YYYY-MM-DD HH:mm:ss`
`--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, execute the `okv backup destination list` command.
`--type` / `type`	Required	Enter one of the following values: • `ONE-TIME` • `PERIODIC`
`--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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "create",
       "options" : {
         "name" : "#VALUE",
         "startTime" : "#NOW|YYYY-MM-DD HH:mm:ss",
         "type" : "#ONE-TIME|PERIODIC",
         "destination" : "#LOCAL|VALUE",
         "interval" : {
           "days" : "#0-99",
           "hours" : "#0-23",
           "mins" : "#0-59"
         }
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-sched-create.json`) and then edit it so that you can create a 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. Execute 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"
   }
   ```

7.11 okv backup schedule delete Command

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, execute the `okv backup schedule list` command. If you need more details about the backup that you want to delete, then execute `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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "delete",
       "options" : {
         "name" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-sched-del.json`) and then edit it so that you can delete the backup schedule.

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "delete",
       "options" : {
         "name" : "okv_backup_hourly"
       }
     }
   }
   ```

3. Execute 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"
   }
   ```

7.12 okv backup schedule get Command

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, execute 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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "get",
       "options" : {
         "name" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-sched-get.json`) and then edit it so that you can get the backup schedule.

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
   ```

```
    "action" : "get",
    "options" : {
      "name" : "okv_backup_hourly"
    }
  }
}
```

3. Execute 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"
  }
}
```

7.13 okv backup schedule list Command

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 input appears as follows:

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "list"
  }
}
```

2. Save the generated input to a file (for example, `bkup-sched-list.json`).
3. Execute the `okv backup schedule list` command using the generated JSON file.

```
okv backup schedule list --from-json bkup-sched-list.json
```

Output similar to the following appears:

```
{
  "result" : "Success",
  "value" : {
    "schedules" : [
      "OKV_BACKUP_HOURLY",
      "LOCAL_BACKUP"
    ]
  }
}
```

7.14 okv backup schedule pause Command

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, execute the `okv backup schedule list` command. If you need more details about the backup that you want to pause, then execute `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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "pause",
       "options" : {
         "name" : "#VALUE"
       }
     }
   }
   ```

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. Execute 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"
   }
   ```

7.15 okv backup schedule resume Command

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, execute the `okv backup schedule list` command. If you need more details about the backup that you want to resume, then execute `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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "resume",
       "options" : {
         "name" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-sched-resume.json`) and then edit it so that you can resume a paused backup schedule.

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "resume",
    "options" : {
      "name" : "okv_backup_hourly"
    }
  }
}
```

3. Execute 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"
   }
   ```

7.16 okv backup schedule update Command

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_in_UTC" ]
[--interval timing]
```

JSON Input File Template

```
{
  "service" : {
    "category" : "backup",
    "resource" : "schedule",
    "action" : "update",
    "options" : {
      "name" : "#VALUE",
      "startTime" : "#NOW|YYYY-MM-DD HH:mm:ss",
      "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, execute the `okv backup schedule list` command. To find the details of the backup that you want to update, execute `okv backup schedule get`.
`--start-time` / `startTime`	Optional	Time to begin the scheduled backup, in double quotation marks. Enter one of the following values, in UTC: • `NOW` • A specific time, using the following format: `YYYY-MM-DD HH:mm:ss`
`--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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "update",
       "options" : {
         "name" : "#VALUE",
         "startTime" : "NOW|#YYYY-MM-DD HH:mm:ss",
         "interval" : {
           "days" : "#0-99",
           "hours" : "#0-23",
           "mins" : "#0-59"
         }
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-sched-update.json`) and then edit it so that you can update the backup schedule.

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "schedule",
       "action" : "update",
       "options" : {
         "name" : "hr_backup",
         "startTime" : "2020-12-20 18:00:00",
         "interval" : {
           "days" : "0",
   ```

```
        "hours" : "12",
        "mins" : "0"
      }
    }
  }
}
```

3. Execute 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"
   }
   ```

7.17 okv backup restore start Command

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 executing the `okv backup restore status` command.

Syntax

```
okv backup restore start --name destination_name --destination
backup_schedule_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 schedule job. To find existing scheduled backups, execute the `okv backup schedule list` command.

Parameter/Template Parameter	Required?	Description
`--destination` / `destination`	Required	Name of the backup destination job. For a listing of existing names, execute the `okv backup destination list` command. If you need more details about the backup that you want to restore, then execute `okv backup schedule get`.

JSON Example

1. Generate JSON input for the `okv backup restore start` command.

   ```
   okv backup restore start --generate-json-input
   ```

   The generated input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "restore",
       "action" : "start",
       "options" : {
         "name" : "#VALUE",
         "destination" : "#VALUE"
       }
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-restore-start.json`) and then edit it so that you can start the restore of a backup schedule. In the following example, the `passphrase` is not specified so that the user will be prompted interactively for it.

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "restore",
       "action" : "start",
       "options" : {
         "name" : "okv_backup_hourly",
         "destination" : "prod_okv_backup_dest"
       }
     }
   }
   ```

3. Execute 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 must be the one that was effective at the time of the backup that is being restored. After you enter this passphrase, output similar to the following appears:

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

7.18 okv backup restore status Command

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 input appears as follows:

   ```
   {
     "service" : {
       "category" : "backup",
       "resource" : "restore",
       "action" : "status"
     }
   }
   ```

2. Save the generated input to a file (for example, `bkup-restore-status.json`).
3. Execute the `okv backup restore status` command using the generated JSON file.

   ```
   okv backup restore status --from-json bkup-restore-status.json
   ```

   If restore is ongoing, then output similar to the following appears:

   ```
   {
     "result" : "Success",
     "value" :
       {
         "destination" : "LOCAL",
         "status" : "ONGOING",
         "time" : "2020-12-18 11:49:30"
   ```

```
    }
}
```

If restore is complete, then output similar to the following appears:

```
{
  "result" : "Success",
  "value" :
    {
      "destination" : "LOCAL",
      "status" : "DONE",
      "time" : "2020-12-18 11:52:02"
    }
}
```

8

Logging, Error Reporting, and Help Information

Logging, error reporting, and finding help information 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.
- Help Information
  You can find information about valid options that the RESTful services utility provides.

8.1 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.

8.1.1 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.

8.1.2 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 8-1 describes the Oracle Key Vault RESTful logging property file parameters.

Table 8-1 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`

Table 8-1 (Cont.) Log Property File Parameters

Logging Property	Description
`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 `%`.
`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`.

8.1.3 Example: Logging File

Oracle Key Vault RESTful services logging files are based on the `java.util.logging` Java logging utility.

Example 8-1 shows a logging file that uses the `INFO` logging level.

Example 8-1 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
```

8.2 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.
- Command Line Error Reporting
  Error reporting captures both faulty actions, such as incorrect passwords, and successful command executions.

8.2.1 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.

8.2.2 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 pfitch --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 pfitch -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::BackupOptio
nBean
[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]
```

8.3 Help Information

You can find information about valid options that the RESTful services utility provides.

For a list of valid options, execute the `okv --help` command. For example:

```
okv --help

Oracle Key Vault REST CLI Version 21.1.0.0.0 Built 11/25/2020 12:56

usage: okv <category> <resource> <action> [--option]
    --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 template file in JSON
    --help List available options
    --okv_client_config <arg> OKV Client configuration
                               file(okvclient.ora) location
    --password <arg> Password
    --profile <arg> Profile name in configuration
                               file(okvrestcli.ini)
    --server <arg> OKV server IP address or hostname
    --user <arg> Username
```

To learn about supported options for a command, you can execute the command with the `--generate-json-input` clause. The output includes the supported command line options for the command. For example, suppose you execute the `okv admin endpoint create` command using the `--generate-json-input` clause:

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

The output indicates the options that you can use when creating an endpoint. In this case, the options are `endpoint`, `description`, `email`, `platform`, `type`, `subgroup`, and `unique`.

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

A.1 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`

Table A-1 (Cont.) Changes to Oracle Key Vault RESTful Commands

Pre-Release 21 Command	Release 21 Command Equivalent
create_unique_endpoint	As an option in `okv admin endpoint create`
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`

Table A-1 (Cont.) Changes to Oracle Key Vault RESTful Commands

Pre-Release 21 Command	Release 21 Command Equivalent
`mod_custom_attr`	`okv managed-object custom-attribute modify`
`modify_endpoint_desc`	An option in `okv admin endpoint update`
`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`

A.2 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.

Commands that cover RESTful services functionality that was not available in releases earlier than Oracle Key Vault release 21.1 are as follows:

- `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

C

D

E

P

R

S

U

V

W