Oracle Transparent Data Encryption and Entrust KeyControl Database Vault

Integration Guide

2024-02-12

Table of Contents

Chapter 1. Introduction

This guide describes the integration of the Entrust KeyControl Database Vault with an Oracle database. KeyControl Database Vault acts as an Extensible Key Management (EKM) solution that securely manages keys and encrypts sensitive data using Transparent Data Encryption (TDE).

- For more detailed information on KeyControl Database Vaults, please refer to the official documentation available at KeyControl Database Vault Documentation.

The Oracle feature Transparent Data Encryption (TDE) provides data-at-rest encryption for sensitive information held by the Oracle database, while at the same time allowing authorized clients to use the database.

Integrated Oracle and Entrust technology has been tested to support Oracle TDE for tablespace encryption, or column encryption, or concurrently for both.

This guides shows support for non-multitenant and multitenant databases. For Oracle version 21C, only multitenant Oracle database types are supported. Oracle does not support the creation of non-multitenant database types on version 21C. For more information on the multitenant support only by Oracle, see the Oracle multitenant documentation.

If using Oracle 18c or later, the `sqlnet.ora` file is officially deprecated and you should use the `WALLET_ROOT` and `TDE_CONFIGURATION` parameters.

1.1. Using this guide

This *Integration Guide* covers UNIX/Linux based systems. It provides:

- An overview of how the Oracle database software and Entrust KeyControl Database Vault work together to enhance security.
- Configuration and installation instructions.
- Depending on your current Oracle setup, how to:
  - Migrate encryption from an existing Oracle wallet or keystore to KeyControl protection.
  - Begin using KeyControl protection immediately if no Oracle software wallet or keystore already exists.

- Examples and advice on how the product may be used.
- Troubleshooting advice.

It is assumed the reader has a good knowledge of Oracle database technology.

Assuming you already have your Oracle database installed, after installing and configuring the Entrust KeyControl Database Vault, there is no other software required. However, some minor configuration changes will be needed.

This guide cannot anticipate all configuration requirements a customer may have. Examples shown in this guide are not exhaustive, and may not necessarily show the simplest or most efficient methods of achieving the required results. The examples should be used to guide integration of the Entrust KeyControl Database Vault with an Oracle database, and should be adapted to your own circumstances.

Entrust accepts no responsibility for loss of data, or services, incurred by use of examples, or any errors in this guide. For your own reassurance, it is recommended you thoroughly check your own solutions in safe test conditions before committing them to a production environment. If you require additional help in setting up your system, contact Entrust Support.

Entrust accepts no responsibility for information in this guide that is made obsolete by changes or upgrades to the Oracle product.

This integration guide assumes that you have already reviewed the documentation for KeyControl Database Vaults and have a basic understanding of the setup processes involved in configuring Oracle database TDE. Familiarity with these concepts will ensure a smoother implementation of the integration.

1.2. Product configuration

Entrust has successfully tested the following software version:

Product	Version	Certification
KeyControl Vault	10.1	FIPS 140 Level 1

Entrust has successfully tested Entrust KeyControl Database Vault with the following configurations:

OS Version	Kernel	Oracle Version
Red Hat Enterprise Linux 8.7 (Ootpa)	Linux 4.18.0-240.el8.x86_64	Oracle Database 21c Enterprise Edition - 21.10.0.0.0
Red Hat Enterprise Linux 8.7 (Ootpa)	Linux 4.18.0-425.10.1.el8_7.x86_64	Oracle Database 19c Enterprise Edition - 19.19.0.0.0

1.3. Conventions used in this document

1.3.1. Multitenant and non-multitenant

Descriptions in this *Integration Guide* may cover non-multitenant databases and multitenant databases. Keep in mind that creation of non-multitenant databases are not supported anymore from Oracle 21C. This guide will use the terms appropriate to the database type under discussion, as outlined:

- Non-multitenant databases are on Oracle version 11g or earlier. Multitenant databases start from Oracle version 12c.
- Non-multitenant database software can only create and use non-multitenant databases. If non-multitenant databases are the subject matter, use the non-multitenant and SQL terminology as shown below.
- Database software supporting multitenant databases may also optionally support non-multitenant databases (pre-21c). In this case, if a non-multitenant mode is the subject matter, then use the non-multitenant terminology and SQL shown below. If a multitenant mode is the subject matter, then use the multitenant terminology and SQL.
- Non-Multitenant (non-container)
    1. Terminology for Oracle software based encryption key repository.

       ALTER SYSTEM SET ENCRYPTION ...

- Multitenant (container)
    1. Terminology for Oracle software based encryption key repository

       Software keystore

    2. SQL preamble for encryption related commands

ADMINISTER KEY MANAGEMENT, etc

Where such terminology applies equally to a software wallet or software keystore, the default terminology software keystore is used to cover both descriptive instances.

1.3.2. Database connections

You must be a user with correct permissions to access a database, and also have the correct privileges to perform the required operations when connected to that database. Your system administrator should be able to create users and grant suitable permissions and privileges according to your organization's security policies.

Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

Example 2

- `<database-user>` is the user identity making the connection.
- `<database-identifier>` is the database to make the connection to.

For the purpose of examples in this guide, the following database users and database identifiers should be sufficient.

- `<database-user>`. This guide will use one following users for connecting to databases:
  - `sysdba`, Oracle's standard sysdba user.
  - `system`, Oracle's standard system user.
  - Non-Mutitenant:
    - `TESTER`, as a local user.
  - Multitenant:
    - `C##TESTER`, as a common user for container (CDB) and the PDBs it contains.
    - `CDB<n>PDB<k>TESTER`, as a local user for a `PDB<k>` within container `CDB<n>`.

      Where `<n>` and `<k>` are distinguishing digits.
- `<database-identifier>`. This guide will use one following database identifies during a connection:

- Non-Multitenant databases:
  - DB, in practice usually the ORACLE_SID of the database. For example:

```
CONNECT sysdba@DB
CONNECT TESTER@DB
```

- Multitenant databases:
  - CDB<n> indicates a container database where <n> is a distinguishing digit.
  - PDB<k> indicates a pluggable database where <k> is a distinguishing digit.

  Multitenant database identifiers will be:

  - CDB<n>, to connect to the $CDB<n>$ROOT for a particular container CDB<n>.
  - CDB<n>PDB<k>, to connect to PDB<k> within CDB<n>.

  For example:

```
CONNECT sysdba@CDB1
CONNECT C##TESTER@CDB1
CONNECT C##TESTER@CDB1PDB2
CONNECT CDB1PDB1TESTER@CDB1PDB1
```

When you are using a multitenant database, the connection implies that you must alter a session if you are not already connected to the required container. For example:

- Example 1:

  CONNECT C##TESTER@CDB<n>

  This implies that, if you are not already connected to CDB<n>, then alter the session:

  ALTER SESSION SET CONTAINER = CDB<n>$ROOT;

- Example 2:

  CONNECT CDB<n>PDB<k>TESTER@CDB<n>PDB<k>

  This implies that, if you are not already connected to CDB<n>PDB<k>, then alter the session:

  ALTER SESSION SET CONTAINER = CDB<n>PDB<k>;

Examples of `sqlplus` connection syntax for different users:

- sqlplus / as sysdba
- sqlplus / as sysdba@CDB1ROOT
- sqlplus CDB1PDB1TESTER/Tester@//localhost:1521/CDB1PDB1.interop.com

1.3.3. Key migration and legacy keys

KeyControl serves as a software wallet or keystore, utilizing the HSM keystore configuration when setting up the wallet type. However, it's important to note that KeyControl is a software-based solution and not a physical hardware security module (HSM).

KeyControl provides two configurations for key management:

- The first configuration entails using pure software-based keys.
- The second configuration utilizes a Hardware Security Module (HSM) as the backend for key creation and operations.

KeyControl offers support for various HSMs, including nShield, Luna, and cloud HSMs.

For more information on HSM configuration with KeyControl, please refer to Hardware Security Modules with KeyControl Vault

Encryption master keys can be migrated between an existing Oracle keystore and KeyControl serving as the wallet. In this case, 'key migration' refers to the transfer of responsibility for holding the master keys.

The encryption keys themselves are not copied or imported between a software keystore and KeyControl wallet Instead, fresh master key(s) are created within the software keystore or KeyControl wallet during the migration. Subsidiary keys that are being protected are re-encrypted using the fresh master key(s). Any new master keys are subsequently created in the current key protector to which you have migrated.

During the re-key process, the previous master keys, or legacy keys, remain in the software keystore or KeyControl wallet where they were originally created. After performing a key migration, you can retain access to the legacy keys in the software keystore or KeyControl wallet you migrated from by setting its passphrase to be the same as the current key protector's passphrase. This allows both the software keystore and KeyControl wallet to be open simultaneously, providing access to the encryption keys they contain. If you do not follow this

approach, you will only be able to access keys in the current key protector. If you are using both a software keystore and KeyControl wallet concurrently, the current key protector is referred to as the primary.

1.4. Overview

Transparent Data Encryption (TDE) is used to encrypt an entire database without requiring changes to existing queries and applications.

When a database encrypted with TDE is loaded into memory from disk storage, it is automatically decrypted, allowing clients to query the database within the server environment without needing to perform any decryption operations. The database is encrypted again when saved to disk storage.

There are several advantages to using KeyControl for managing Transparent Data Encryption (TDE) within the Oracle environment. Firstly, it increases visibility into TDE keys, providing administrators with better oversight and control. KeyControl supports the use of in-house Hardware Security Modules (HSMs) for generating cryptographic material, ensuring a secure and trusted key management process. Administrators also have granular control over TDE key usage, with the ability to revoke access if database keys are suspected to be compromised.

Furthermore, KeyControl provides the advantage of storing keys externally to the Oracle Server, offering an additional layer of protection. Access control is strengthened through the validation of the Oracle Server VM's certificate by KeyControl, enhancing overall security. Encryptions keys are securely stored on a FIPS 140 Level 1 certified Encrypted Object store, ensuring compliance with stringent security standards.

KeyControl also enables geo-location-based access control when boundary control is enabled, allowing for fine-grained access restrictions based on geographical locations. Additionally, audit logs are generated in KeyControl, providing a comprehensive record of key management activities for compliance and auditing purposes. Overall, leveraging KeyControl for TDE management enhances security, control, and compliance within the Oracle environment.

Chapter 2. Procedures

Steps:

1. Preparatory requirements
2. Create a Database Vault in the KeyControl Vault Server
3. Downloading Policy Agent
4. Create a VMSet in KeyControl Vault
5. Create Key Set
6. Link PKCS11 Library
7. Opening and closing a keystore or KeyControl
8. Migrating from software wallet to KeyControl (non-multitenant)
9. Migrating from software keystore to KeyControl (multitenant)
10. Create master keys directly in KeyControl for non-multitenant database
11. Create master keys directly in KeyControl for multitenant database
12. Rekeying or key rotation
13. Enabling and Disabling Database Connector

2.1. Preparatory requirements

Before installing the software, Entrust recommends that you familiarize yourself with:

- The Oracle database TDE documentation and setup process.
- The Entrust KeyControl Vault documentation.
- Entrust recommends that you create a policy for managing SQL scripts that allow use of credentials for the Oracle database. These SQL scripts should only be available to authorized users.

This guide assumes that Oracle database software, and (at least) one Oracle database, is already installed on your system. With Oracle database software already installed, ensure that any required patches have been added.

To integrate an Oracle database with Entrust KeyControl Database Vault, the following steps are required:

1. Environment configuration.
2. Install the Entrust KeyControl Database Vault software.

3. Configure Oracle database software to use the Entrust KeyControl Database Vault.

Details of your installation and configuration will depend on:

- Whether you are using a non-multitenant or multitenant database.
- Whether you want to migrate encryption keys from an existing Oracle software keystore to Entrust KeyControl, or start directly withEntrust KeyControl.

The default host server user is `oracle` unless stated otherwise.

For more information on how to configure your Entrust environment, refer to the KeyControl Vault Installation Guide.

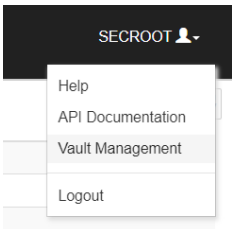
For more information on how to configure your Oracle environment, see the Oracle documentation.

2.2. Create a Database Vault in the KeyControl Vault Server

The KeyControl Vault appliance supports different type of vaults that can be used by all type of applications. This section describes how to create a Database Vault in the KeyControl Vault Server.

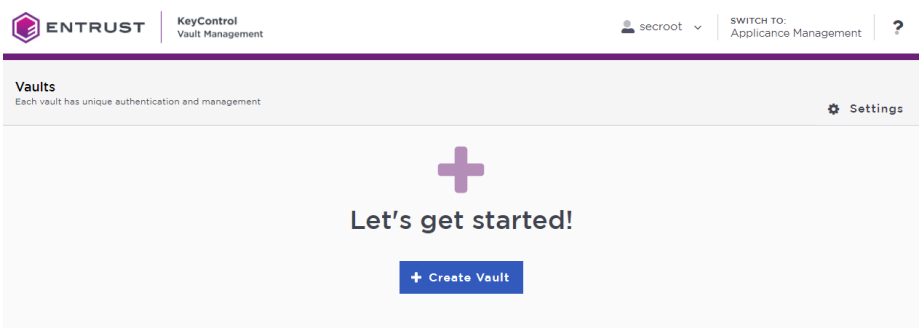
Please refer to Creating a Vault section of the admin guide for more details about it.

1. Log into the KeyControl Vault Server web user interface:
   a. Use your browser to access the IP address of the server.
   b. Sign in using the **secroot** credentials.
2. Select the user's drop-down menu and select **Vault Management**.



   This action will take you to the KeyControl Vault Management interface.

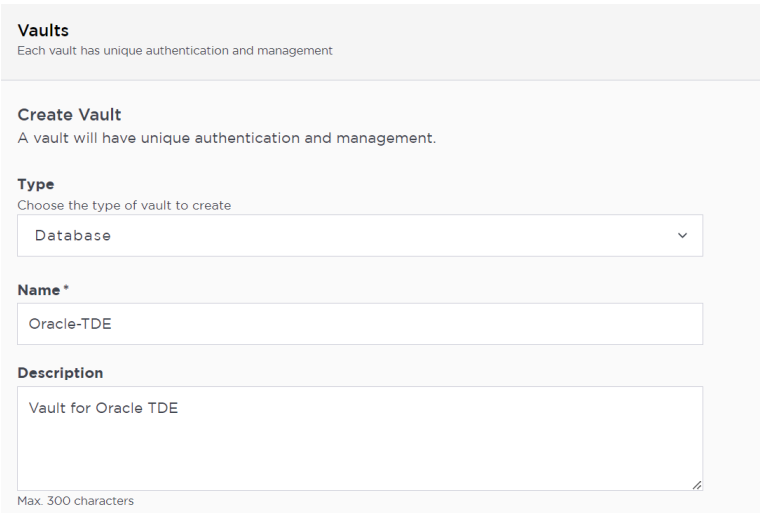
3. In the KeyControl Vault Management interface, select **Create Vault**.

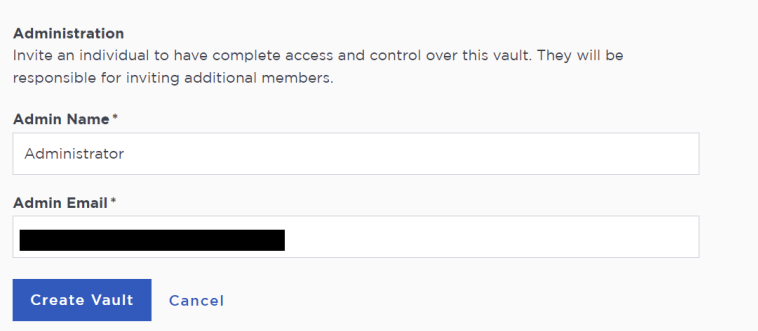


KeyControl Vault supports the following types of vaults:

- **Cloud Key Management** - Vault for cloud keys such as BYOK and HYOK.
- **KMIP** - Vault for KMIP Objects.
- **PASM** - Vault for objects such as passwords, files, ssh keys, and so on.
- **Database** - Vault for database keys.
- **Tokenization** - Vault for tokenization policies.
- **VM Encryption** - Vault for encrypting VMs.

4. In the **Create Vault** page, create a **Database** Vault:
   a. For **Type**, select **Database**.
   b. For **Name**, enter the name of the vault
   c. For **Description**, enter the description of the vault.
   d. For **Admin Name**, enter the name of the administrator of the vault.
   e. For **Admin Email**, enter a valid email for the administrator.

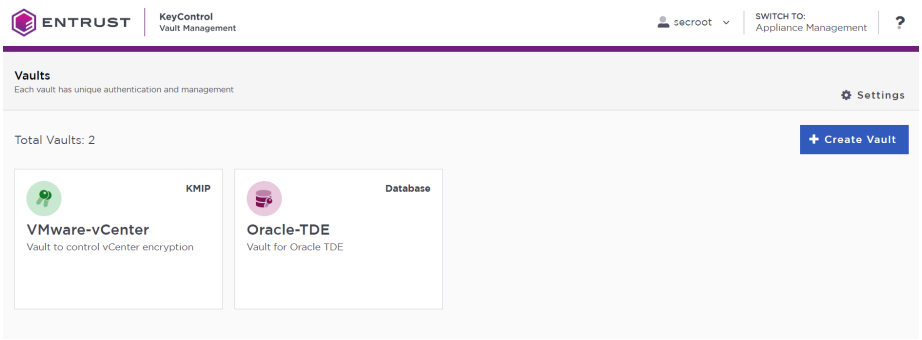




A temporary password will be emailed to the administrator's email address. This is the password that will be used to login for the first time to the Database Vault space in KeyControl.

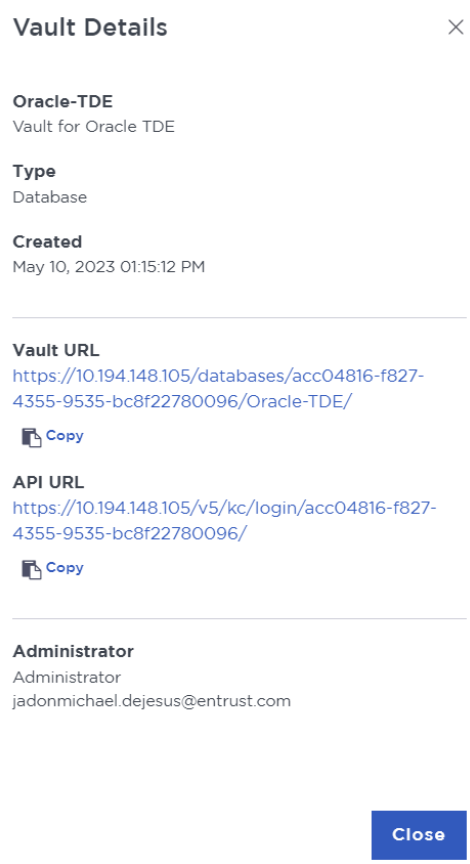
5. Select **Create Vault**.
6. Select **Close** when the vault creation completes.

   The newly-created vault is displayed in the Vault dashboard. For example:



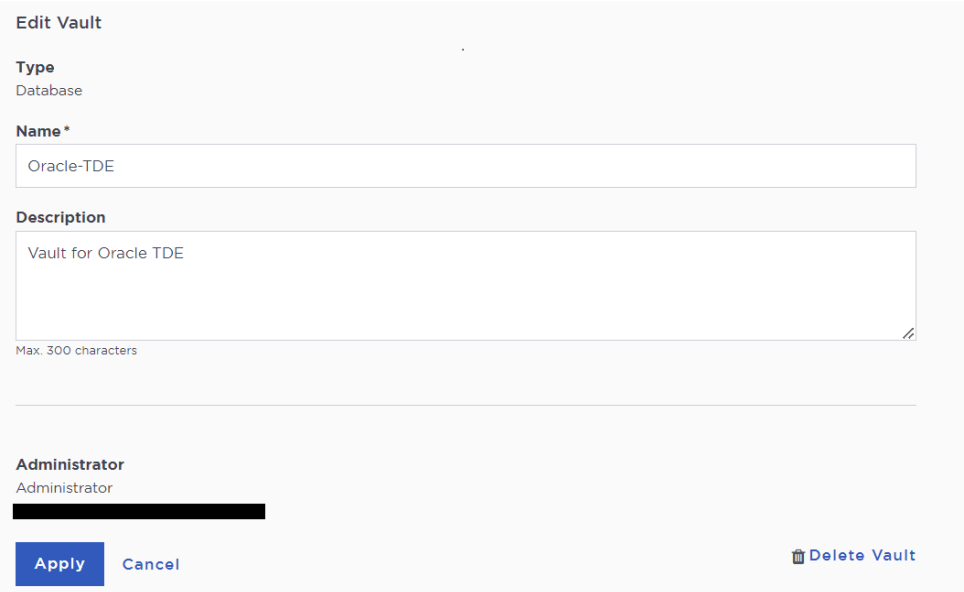
2.2.1. View Vault's Details

To view the details on the vault, select **View Details** when you hover over the vault.



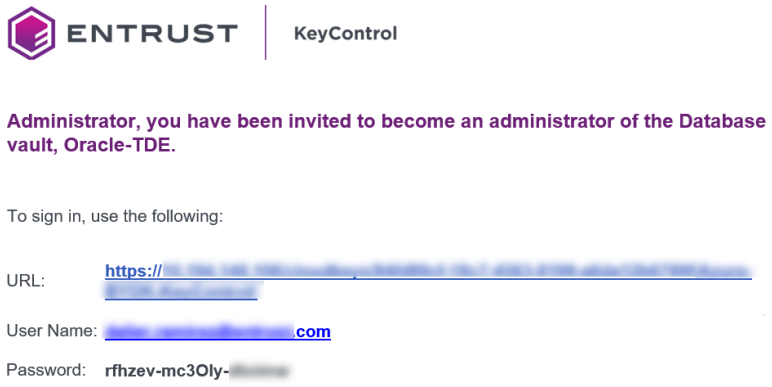
2.2.2. Edit Vault

To edit the details of the vault, select **Edit** when you hover over the vault.



2.2.3. Managing the Vault

After the Vault has been created, look for the email that was sent with the Vault's URL and the login information for the Vault. For example:



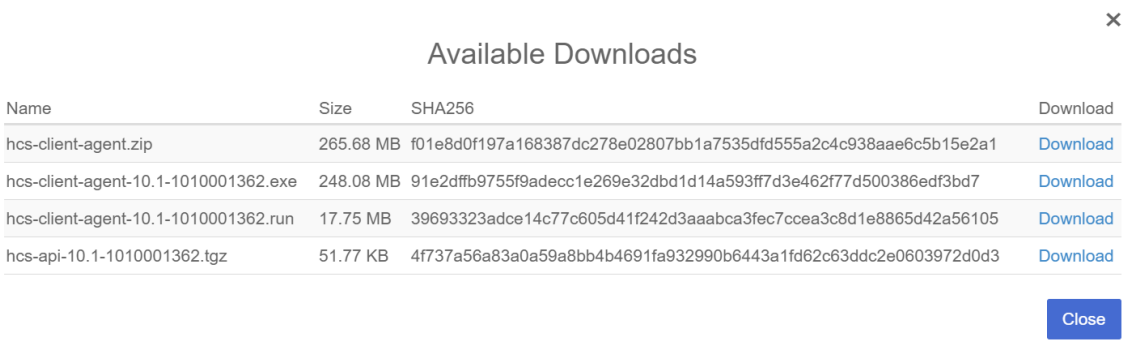
Go to the URL and login with the credentials given. When you login for the first time, the system will ask the user to change the password.

2.3. Downloading Policy Agent

The nShield DataControl Policy Agent serves as a software module that facilitates encryption of virtual disks and individual files on Windows and Linux operating systems, enabling secure sharing of encrypted data among VMs. When a user attempts to access an encrypted disk, the Policy Agent ensures authorization by verifying the request with KeyControl Vault. Furthermore, the configuration of the Policy Agent includes the setup of the Oracle server to load the EKM provider library, reinforcing the encryption capabilities within the server environment.

1. Log into the newly created vault:
    a. Select the **WORKLOADS** tab.
    b. Select **Actions** > **Download Policy Agent**

A list of available downloads appears. For example:



2. Download the `hcs-client-agent-10.1-1010001362.run` file.
3. Return to the Oracle server as `root` user and install `pkcs11-tool` for testing:

```
yum install opensc
```

4. Check if Python is already installed:

```
python --version
```

5. If Python is already installed, it will display the version number. If not, you can install it using the package manager for your Linux distribution:

```
sudo dnf install python3
```

6. After installing Python, check that it is in your system path:

```
python --version
```

7. Navigate to the directory where you downloaded the `hcs-client-agent-10.1-1010001362.run` file.
8. Make the file executable:

```
chmod +x hcs-client-agent-10.1-1010001362.run
```

9. Run the installer:

```
./hcs-client-agent-10.1-1010001362.run
```

For example:

```
% ./hcs-client-agent-10.1-1010001362.run

Verifying archive integrity...  100%   All good.
Uncompressing hcs-client-agent-10.1-1010001362.run  100%
x86_64
No HyTrust Agent found on this system
HyTrust Agent will be installed in /opt/hcs
Specify location for installing HyTrust Agent (/opt/hcs):
Created symlink /etc/systemd/system/multi-user.target.wants/hcld.service →
/usr/lib/systemd/system/hcld.service.
Platform is rhel

You can now install online encryption driver, the process is described in the Admin Guide
Please see the following section of Admin Guide for details
--- Administration Guide > Data Encryption > Linux Encryption Overview
```

```
Installation successful
```

10. Verify the installation:

```
% hcl status
```

For example:

```
% hcl status

Summary
--------------------------------------------------------------------------------
KeyControl: None
Status: Not registered
AES_NI: enabled
HTCRYPT: Not Installed

Registered Devices
--------------------------------------------------------------------------------
Disk Name          Cipher       Status                  Clear
--------------------------------------------------------------------------------

Available Devices
--------------------------------------------------------------------------------
Disk Name            Device Node                    Size (in MB)
--------------------------------------------------------------------------------

...
```

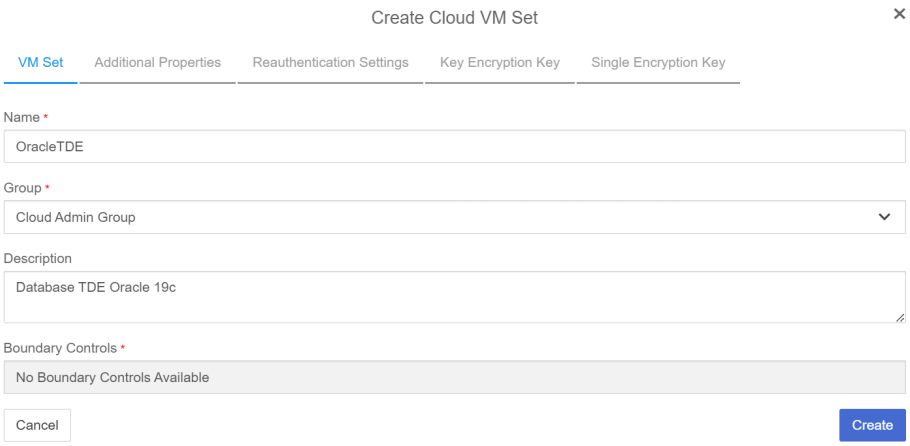
For more information, please refer to Entrust DataControl Policy Agent

2.4. Create a VMSet in KeyControl Vault

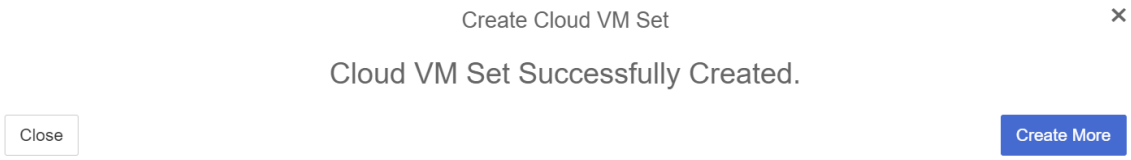
A VMSet in KeyControl Vault is a logical grouping of virtual machines (VMs) that allows for centralized management and control of encryption policies.

1. Login to the KeyControl Database Vault for Oracle TDE.
2. Select the **WORKLOADS** tab.
3. Select **Actions** > **Create New Cloud VM Set**.
4. In the **Create Cloud VM Set** page:
    a. Enter a **Name** for the cloud VM set
    b. For **Group**, select **Cloud Admin Group**
    c. Enter a **Description**
    d. Select **No Boundary Controls Available**

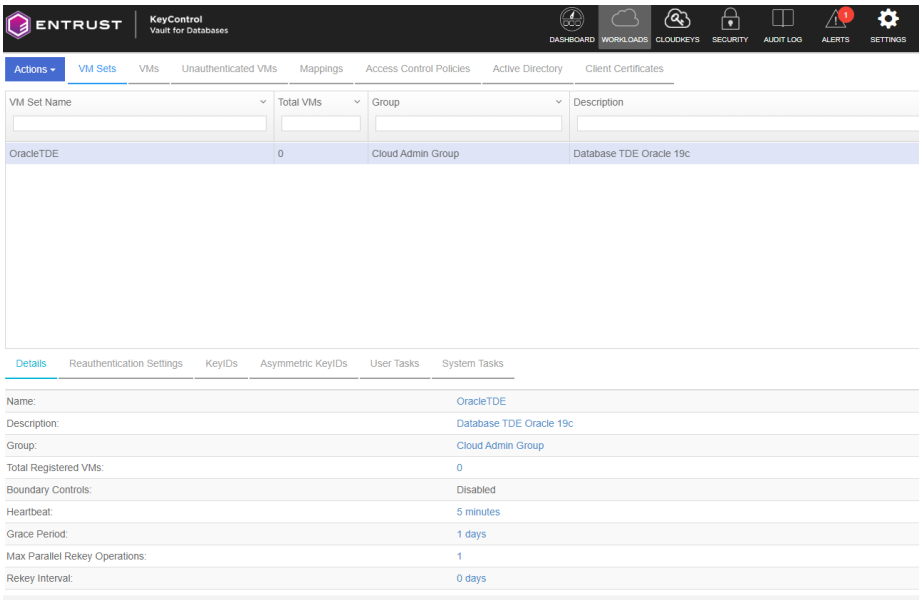
For example:



5. Select **Create**.
6. When a success message appears, click **Close**.



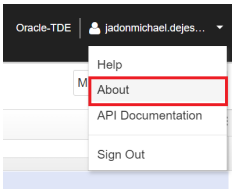
7. The newly-created VM Set is added to the list. For example:



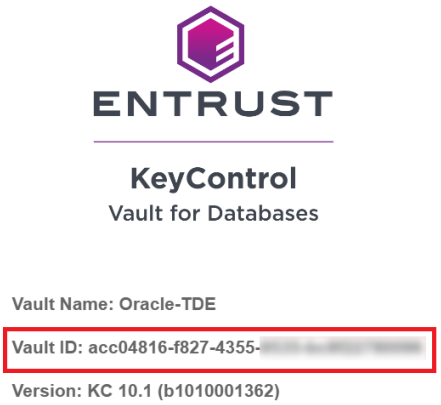
2.4.1. Register the Oracle Server VM to VM Set

To register the Oracle Server VM to VM Set:

1. In the vault page, click on your user in the top corner and select **About**.



2. Copy the Vault ID. For example:



3. Register KeyControl in the Oracle Server:

```
% hcl register -a -v <VAULT-ID> <KEYCONTROL-VAULT-IP>
```

Enter the number corresponding to the VM created earlier, then enter y to confirm the VM and continue. For example:

```
% hcl register -a -v <VAULT-ID> <KEYCONTROL-VAULT-IP>

Please provide the Vault login details
username: user@entrust.com
password: ********

Available Cloud VM Sets
-------------------------------------------------------------------------------
1 : OracleTDE
-------------------------------------------------------------------------------

Please select a Cloud VM Set by number to which this VM should be added:
!Error: Not a valid selection


Available Cloud VM Sets
-------------------------------------------------------------------------------
1 : OracleTDE
-------------------------------------------------------------------------------

Please select a Cloud VM Set by number to which this VM should be added: 1

The selected Cloud VM Set is -- OracleTDE
Do you want to continue (y/n)?y
Registered as otde-19c-kc10.1-redhat-8 with KeyControl node(s) **.***.***.***

Completing authentication for otde-19c-kc10.1-redhat-8 on KeyControl node(s) **.***.***.***
```

```
Authentication complete, machine ready to use
Getting KeyControl Mapping information

KeyControl Mappings are not available
```

4. Verify the registration:

```
% hcl status
```

For example:

```
% hcl status
Summary
--------------------------------------------------------------------------------
KeyControl: [IP]
KeyControl list: [IP]
Vault ID: [ID]
Status: Connected
Last heartbeat: Wed May 10 15:16:39 2023 (successful)
AES_NI: enabled
Certificate Expiration: May  9 19:16:15 2024 GMT
HTCRYPT: Not Installed

Registered Devices
--------------------------------------------------------------------------------
Disk Name          Cipher       Status                  Clear
--------------------------------------------------------------------------------

Available Devices
--------------------------------------------------------------------------------
Disk Name            Device Node                    Size (in MB)
--------------------------------------------------------------------------------

...
```

5. Enable TDE:

```
hcl tde enable
```

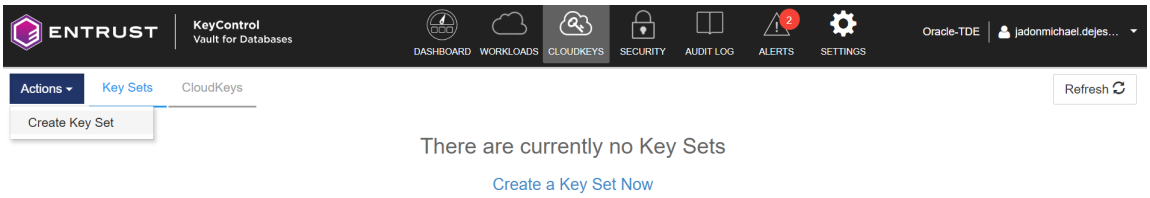
For example:

```
% hcl tde status
TDE is  not enabled on this VM

% hcl tde enable
Enabling tde will change permissions of some Files.
Do you want to proceed? (y/n) y

% hcl tde status
TDE is enabled on this VM
```

2.5. Create Key Set

A KeySet in KeyControl Vault serves as a container for managing encryption keys used in various cryptographic operations.

1. Login to the KeyControl Database Vault.
2. Select the **CLOUD KEYS** and then select the **Key Sets** tab.
3. Select **Actions** > **Create Key Set**. For example:

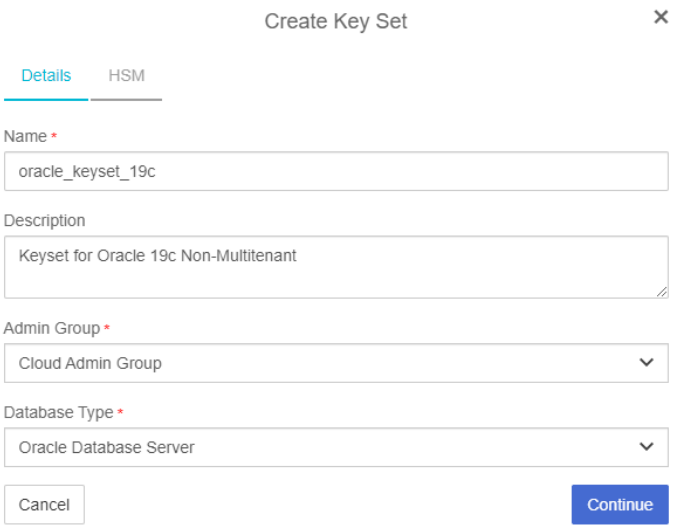


   The **Create Key Set** dialog appears.

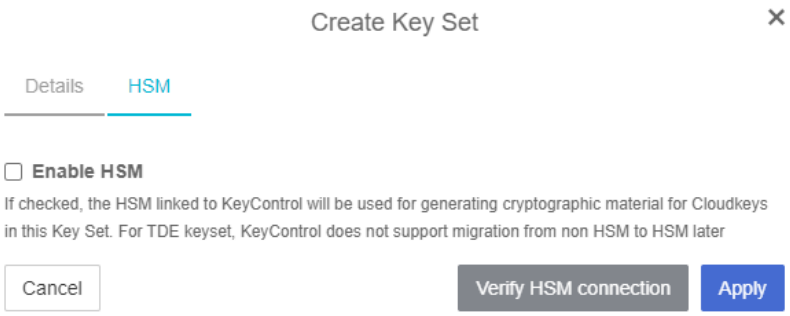
4. In the **Details** tab, create the Key Set:

   a. Enter a **Name**.
   b. Enter a **Description**.
   c. For **Admin Group**, select **Cloud Admin Group**.
   d. For **Database Type**, select **Oracle Database Server**.

   For example:



5. Select **Continue**.
6. In the **HSM** tab, clear the **Enable HSM** check box. The KeySet can also be created with HSM enabled, providing administrators with the ability to safeguard the TDE master keys using an HSM. However, prior to creating the KeySet, the HSM must be properly configured within KeyControl.



For more information on HSM configuration with KeyControl, please refer to Hardware Security Modules with KeyControl Vault

7. Select **Apply**.

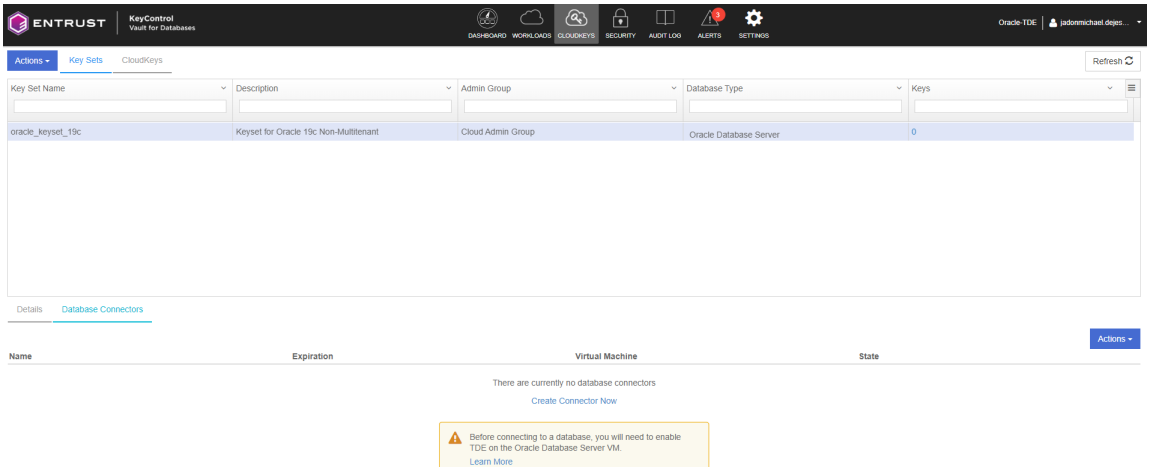
   Click **Close** when a success message appears.

8. Verify that your KeySet is listed.

2.5.1. Create Database Connector

The Database Connector creates a connection between the KeySet and the registered VM, enabling secure communication. Access credentials are associated with the connector, providing authentication for data access. The connector also allows for controlled access, empowering the controller to manage privileges effectively.

1. Select the newly created KeySet > Select the Database tab > Select **Create Connector Now**



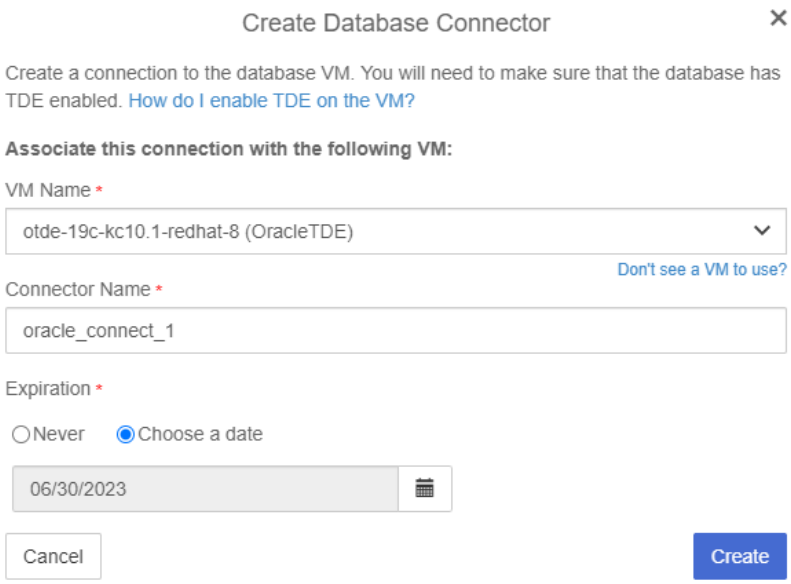
2. Create Database Connector:

   a. Select the **VM Name**

b. Enter a **Connector Name**

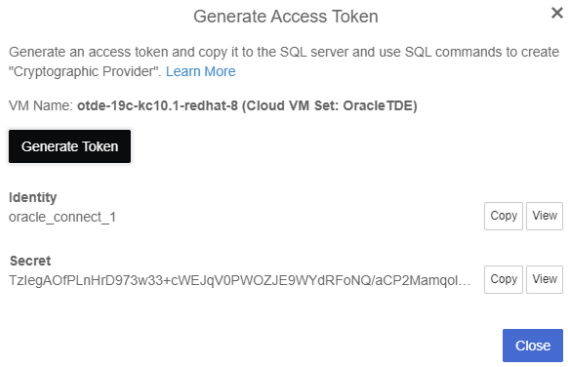
c. Select an **Expiration**

d. Select **Create**



3. Select the newly created database connector > **Actions** > **Generate Access Token**

4. Select **Generate Token** and it will display the newly generated token.
5. Copy the access token (**Identity** and **Secret**). For example:



6. In your Oracle Server, create a config file `/opt/oracle/entrust/orcl.conf` using the copied Access Token (Identity and Secret) that will be used by the database administrators.

   The `orcl.conf` file must be in a JSON format. For example:

```
{
  "identity": "oracle_connect_1",
  "secret":
"TzIegAOfPLnHrD973w33+cWEJqV0PWOZJE9WYdRFoNQ/aCP2MamqolkM2lU2qgLRo9WR+9u2EcVA3w0Dh5tXsyP4QAiyo8Mbe2W3bBz2Fx
AL7Z/IE/9l5gDcuQQ+riJ+EkTWvnCqd660NvNWhWkXCXzqAYk7ObicE0IazEqSxEfF8/reKuJcSDd0/+gMdbJ2Uk0c4r2VI58S7dp0CMA5D
L7CGLppbKwcONq0q0uNa5/AvQd9oYmXCGDzcfinZzAkNJ2yIaDIerLa+ME8gSPtyv4hh+7RtGM+uHnB2TD7MSz88l46nBBxK33u+eXyZBRb
Dis9K0supoWXoEhgoEV3P5HS9+Hz3KCnQA4PnydgcLqnUbNYZqAboShPSkEHFHQkV2hW5/9tEz3MMaiHCJHT7Tx9hXTsyQpQ6B8O+CqPSHE
2Ba0NHTBkdUiu/dnuWJSNTNblvIsrT95lKKIgyCf0VWKH9HFRWmE9DwmZo85Tv0oeCCNbcFlljWTC/hFOGsiBTpcCoQy/KG0oGyy1y/o+Zt
CR3O7sW7Hbr4gh0mkXP8ae+JSm2BV4eSACwAKlsYQlh1MqsRrUtJyZhpUpdc9qai7O9u0KwzrGaecN229zQVAHb4VUpiJm9NXnyram46P/S
K6MpooPEcHFIVriPfkiPrxdILrKYu2W9bXI53FVtbM="
}
```

7. Set the following ownership and permissions on the `/opt/oracle/entrust` directory:
    - **Owner:** oracle
    - **Group:** oinstall
    - **Permissions:** 775

```
mkdir /opt/oracle/entrust
sudo chown oracle:oinstall /opt/oracle/entrust
sudo chmod 775 /opt/oracle/entrust
```

The access credentials will be securely stored on the Oracle server, enabling the creation and utilization of the master key. By leveraging these credentials, you gain the ability to enable robust encryption on the database, making use of the master key for enhanced security.

For more information on creating a KeyControl Vault Key Set for TDE, please refer to KeyControl Vault Key Set for TDE

2.6. Link PKCS11 Library

You must now configure the Oracle PKCS#11 Library folder to use the nShield KeyControl PKCS#11 API.

1. Create a directory path for the nShield API library as the `oracle` user. Make ownership and permissions on the directory as:
    - **Owner:** oracle
    - **Group:** oinstall
    - **Permissions:** 775

```
#
# ORACLE_BASE is typically /opt/oracle
#
sudo chown -R oracle:oinstall $ORACLE_BASE
```

```
sudo chmod -R 775 $ORACLE_BASE
mkdir -p $ORACLE_BASE/extapi/64/hsm/entrust
chown oracle:oinstall $ORACLE_BASE/extapi/64/hsm/entrust
chmod 775 $ORACLE_BASE/extapi/64/hsm/entrust
```

2. Link the PKCS#11 Library into the directory as the `oracle` user:

```
ln -s /opt/hcs/lib/libpkcs11.so $ORACLE_BASE/extapi/64/hsm/entrust/libpkcs11.so
```

2.7. Testing the integration

When testing the integration, make sure you use the instructions that are appropriate to your installation. That is:

- A non-multitenant database and software wallet.
- A multitenant database and software keystore.

For this integration, it is necessary to update the PKCS#11 library for Oracle 19c and 21c Multitenant Database. Failure to do so may lead to the following issues:

- Failing at migrating the software wallet to KeyControl.
- Failing at opening KeyControl keystore for all containers.

For example:

```
ERROR at line 1:
ORA-28407: Hardware Security Module failed with PKCS#11 error
CKR_SESSION_HANDLE_INVALID(179)

ERROR at line 1:
ORA-03113: end-of-file on communication channel
Process ID: 148627
Session ID: 379 Serial number: 61809
```

For the most accurate and updated information regarding the PKCS11 library fix, nShield recommends referring to the *KeyControl 10.1 Release Notes*.

2.8. Opening and closing a keystore or KeyControl

Oracle has a control system that gates access to a software keystore or KeyControl:

- If a keystore or wallet is open, then you can access its contents.
- If a keystore or wallet is closed, then you cannot access its contents.

You can open or close a software keystore or wallet with the following SQL statements.

2.8.1. Non-multitenant

This section assumes database is open.

CONNECT TESTER@DB or CONNECT sysdba@DB

- To open a wallet:

```
ALTER SYSTEM SET [ENCRYPTION] WALLET OPEN IDENTIFIED BY "<credential>";
```

- To close a wallet, pre-11.2.0.1.0:

```
ALTER SYSTEM SET [ENCRYPTION] WALLET CLOSE;
```

- To close a wallet, 11.2.0.1.0 or later:

```
ALTER SYSTEM SET [ENCRYPTION] WALLET CLOSE IDENTIFIED BY "<credential>";
```

In all of the above, the `[ENCRYPTION]` clause is optional.

2.8.2. Multitenant

This section assumes the respective CDB and PDB databases are open:

- To open/close a keystore for the container (CDB) only:

  CONNECT C##TESTER@CDB<n>

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<credential>";
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "<credential>";
```

- To open/close a keystore for the container (CDB) and all PDBs it holds:

  CONNECT C##TESTER@CDB<n>

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<credential>" CONTAINER=ALL;
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "<credential>" CONTAINER=ALL;
```

  If you want to close all keystores, use the following SQL:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE CONTAINER=ALL;
```

- To open/close a keystore for a single PDB, you must use same credential as used by the containing CDB.

  CONNECT PDB<k>TESTER@CDB<n>PDB<k>

  ```
  ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "<credential>";
  ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "<credential>";
  ```

  During migration from Software Wallet to the KeyControl Keystore, you may experience issues closing the keystore. To resolve this, disable the auto-login keystore to close all keystores. See How To Disable Auto-Login Keystore for full details.

  ```
  sudo -u oracle mv <path-to-keystorefolder>/<keystore-folder>/tde/cwallet.sso <path-to-
  keystorefolder>/<keystore-folder>/tde/cwallet.sso.backup
  ```

2.9. Migrating from software wallet to KeyControl (non-multitenant)

The following procedure applies when the target database is non-multitenant, and you are already using a software wallet with TDE encryption. If your target database is multitenant, see Migrating from software keystore to KeyControl (multitenant).

Entrust strongly recommends you back up your software wallet as an independent operation before attempting migration to KeyControl. Keep the backup folder in a safe place separated from the associated database files. Only users with authorization should be able to access the backup folder.

Repeat the following procedure for each database software wallet from which you want to migrate. Each independent database instance can use its own Entrust key protection method or credential if required.

Once an Entrust key protection method has been activated for a particular database instance, then you must continue to use that same credential for any further keys you want to protect for that instance.

Use the `WALLET_ROOT` and `TDE_CONFIGURATION` parameters. It is assumed the `WALLET_ROOT` parameter has already been set for Oracle keystore use.

In the following steps, use the `orcl.conf` file to utilize the access credentials for the KeyControl Database Vault.

1. Prepare for key migration by running the following SQL script:

   CONNECT sysdba@DB

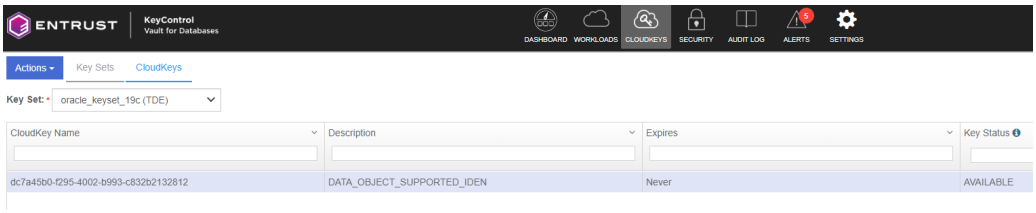
   ```
   ALTER SYSTEM SET TDE_CONFIGURATION = "KEYSTORE_CONFIGURATION=HSM|FILE" SCOPE=BOTH SID='*';
   ```

2. Migrate from the keystore to KeyControl:

   CONNECT sysdba@DB

   ```
   ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" MIGRATE USING <keystore-passphrase>;
   ```

   The cloud key is created. For example:

3. Return to the Oracle Server in the SQL database and run the following command:

   ```
   select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;
   ```

   For example:

   ```
   SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;

       CON_ID WRL_TYPE            STATUS
   ---------- -------------------- ------------------------------
            0 HSM                  OPEN
   ```

2.10. Migrating from software keystore to KeyControl (multitenant)

The following procedure applies when the target database is multitenant, and you are already using a software wallet with TDE encryption. If your target database is non-multitenant, see Migrating from software wallet to KeyControl (non-multitenant).

Repeat the following procedure for each software keystore from which you want to migrate. Each container database (CDB) can use its own Entrust key protection method (credential) if required. However, once a Entrust key protection method has been activated for a particular database instance (CDB), then you must continue to use that same credential for any further keys you want to protect for that instance.

Use the `WALLET_ROOT` and `TDE_CONFIGURATION` parameters.

In the following steps, use the `orcl.conf` file to utilize the access credentials for the KeyControl Database Vault.

1. Back up your software keystore before attempting key migration to KeyControl:

   CONNECT sysdba@CDB<n>

   ```
   ADMINISTER KEY MANAGEMENT BACKUP KEYSTORE USING '<PreMigrationBackupString>' IDENTIFIED BY
   "<keystorepassphrase>";
   ```

2. Prepare for key migration by running the following SQL script:

   CONNECT sysdba@CDB1ROOT

   ```
   ALTER SYSTEM SET TDE_CONFIGURATION = "KEYSTORE_CONFIGURATION=HSM|FILE" SCOPE=BOTH SID='*';
   ```

3. Migrate from the keystore to KeyControl:

   CONNECT sysdba@CDB1ROOT

   ```
   ALTER SESSION SET CONTAINER = CDB$ROOT;
   SHOW con_name;

   --Open all the PDBs.
   ALTER PLUGGABLE DATABASE ALL OPEN;

   ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" MIGRATE
   USING <keystore-passphrase> WITH BACKUP;
   ```

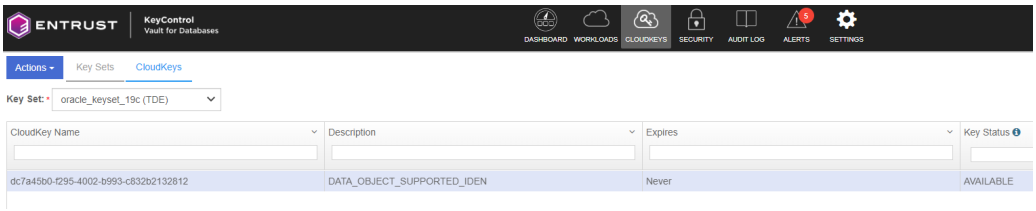
4. For Oracle 21c Database, you must open PDB$SEED with read write permissions to successfully bounce the database after migrating from the keystore to KeyControl:

   CONNECT sysdba@CDB1ROOT

   ```
   ALTER SESSION SET CONTAINER = CDB$ROOT;
   SHOW con_name;
   ```

```
--Open all the PDBs.
ALTER PLUGGABLE DATABASE ALL OPEN;

-- open PDB$SEED with read write perm show pdbs;
alter pluggable database pdb$seed close;
alter pluggable database pdb$seed open read write;
show pdbs;

ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" MIGRATE
USING <keystore-passphrase> WITH BACKUP;
```

For example:



5. Return to the Oracle Server in the SQL database and run the following command:

```
select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;
```

For example:

```
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;

    CON_ID WRL_TYPE             STATUS
---------- -------------------- ------------------------------
         1 HSM                  OPEN
         2 HSM                  OPEN
         3 HSM                  OPEN
         4 HSM                  OPEN
```

2.11. Create master keys directly in KeyControl for non-multitenant database

The following procedure applies when the target database is non-multitenant, and there is no pre-existing software wallet. If your target database is multitenant, see Create master keys directly in KeyControl for multitenant database.

Repeat the following procedure for each database in which you want to create keys. Each database can use its own Entrust key protection method (credential) if required. However, once an Entrust key protection method has been activated for a particular database instance, then you must continue to use that same credential for any further keys you want to protect for that instance.

2.11.1. Use the WALLET_ROOT and TDE_CONFIGURATION parameters

1. Set up the `WALLET_ROOT` and `TDE_CONFIGURATION` parameters as follows. You must set up the `WALLET_ROOT` parameter even if you do not use a keystore.

   CONNECT sysdba@DB

   ```
   ALTER SYSTEM SET WALLET_ROOT = "/opt/oracle/entrust" scope=SPFILE;
   ```

2. Bounce the database after setting up the `WALLET_ROOT` parameter.

   CONNECT sysdba@DB

   ```
   ALTER SYSTEM SET TDE_CONFIGURATION = "KEYSTORE_CONFIGURATION=HSM" SCOPE=BOTH SID='*';
   ```

3. Bounce the database after setting up the `TDE_CONFIGURATION` parameter.

2.11.2. Create the encryption keys

In the following steps, use the `orcl.conf` file to utilize the access credentials for the KeyControl Database Vault to enable the creation and utilization of the master key.

1. Select the protection method (credential) that you require below, and run the SQL.

   CONNECT TESTER@DB or CONNECT sysdba@DB

   ```
   ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf";
   ```

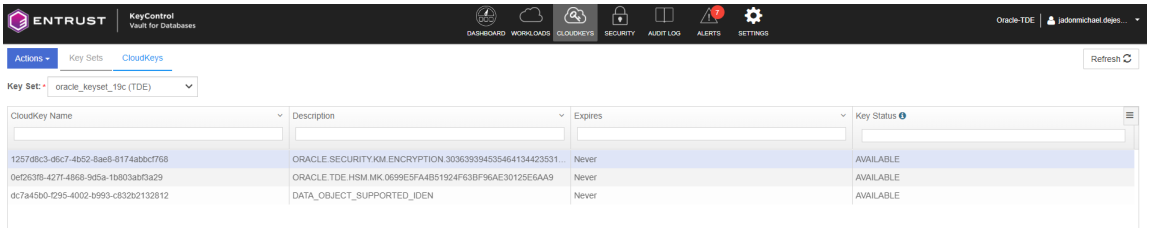
   Run this command to see the status change:

   ```
   select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;
   ```

   For example:

   ```
   SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;

       CON_ID WRL_TYPE            STATUS
   ---------- -------------------- ------------------------------
            0 HSM                  OPEN
   ```

   The encryption key is created. For example:



After you created the master encryption keys in KeyControl as above, proceed to encrypt your database by using tablespace encryption, column encryption, or both, as usual.

2.12. Create master keys directly in KeyControl for multitenant database

The following procedure applies when the target database is multitenant, and there is no preexisting software keystore. If your target database is non-multitenant, see Create master keys directly in KeyControl for non-multitenant database.

Repeat the following procedure for each database in which you want to create keys. Each database instance can use its own Entrust key protection method (credential) if required. However, once an Entrust key protection method has been activated for a particular database instance (CDB), then you must continue to use that same credential for any further keys you want to protect for that instance.

You must create the container (CDB) master key first. After the CDB master key has been created you have a choice of creating master keys for all the PDBs it contains in one operation, or else for each PDB individually.

The PDB(s) must use the same protection credential as the CDB.

2.12.1. Use the WALLET_ROOT and TDE_CONFIGURATION parameters

To set and use the WALLET_ROOT and TDE_CONFIGURATION parameters:

1. Set up the `WALLET_ROOT` and `TDE_CONFIGURATION` parameters as follows. You must set up the `WALLET_ROOT` parameter even if you do not use a keystore.

   CONNECT sysdba@CDB1ROOT

   ```
   ALTER SYSTEM SET WALLET_ROOT = "/opt/oracle/entrust" scope=SPFILE;
   ```

2. Bounce the database after setting up the `WALLET_ROOT` parameter.
3. Run the following command:

```
ALTER SYSTEM SET TDE_CONFIGURATION = "KEYSTORE_CONFIGURATION=HSM" SCOPE=BOTH SID='*';
```

4. Bounce the database after setting up the `TDE_CONFIGURATION` parameter.

2.12.2. Create the CDB and then all PDB master keys in one operation

In the following steps, use the `orcl.conf` file to utilize the access credentials for the KeyControl Database Vault to enable the creation and utilization of the master key.

1. Run the following SQL:

   CONNECT C##TESTER@CDB<n>

```
ALTER SESSION SET CONTAINER = CDB$ROOT;
SHOW con_name;

ALTER DATABASE OPEN;
ALTER PLUGGABLE DATABASE ALL OPEN READ WRITE;

ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" CONTAINER =
ALL;
```

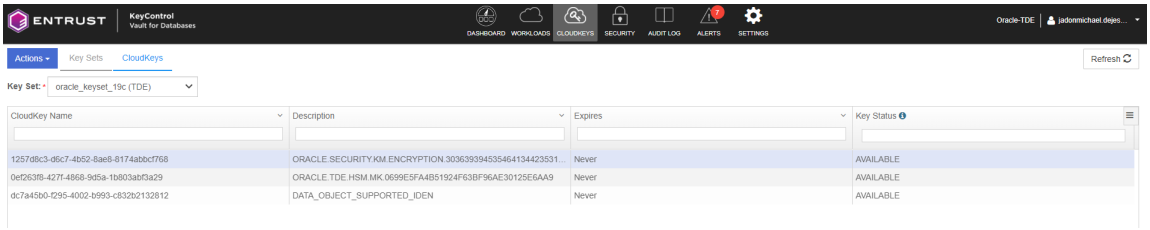
2. Activate master keys for the CDB and all the PDBs in one operation:

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" WITH BACKUP CONTAINER
= ALL;
```

   Run this command to see the status change:

```
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;

    CON_ID WRL_TYPE             STATUS
---------- -------------------- ------------------------------
         1 HSM                  OPEN
         2 HSM                  OPEN
         3 HSM                  OPEN
         4 HSM                  OPEN
```

   The master key is created. For example:



Encrypt your database using tablespace encryption, column encryption, or both.

2.12.3. Create the CDB master key and a single PDB master key

To create the CDB master key and a single PDB master key:

1. Create the CDB master key:

   CONNECT C##TESTER@CDB<n>

   Select the protection method you require below, and run the SQL:

   ```
   --This will activate the credential if it isn't already
   ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf";
   ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" WITH BACKUP;
   ```

2. Repeat the following step to create a single PDB master key for any PDB you select.

   CONNECT PDB<k>TESTER@CDB<n>PDB<k>

   You must use the same protection method (credential) as the containing CDB.

   Run the SQL:

   ```
   --If the PDB is already open, you don't need to do this.
   ALTER PLUGGABLE DATABASE <CDB<n>PDB<k>> OPEN READ WRITE;

   --If the keystore is already open, you don't need to do this.
   ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf";

   --Make the master key for the PDB you should be currently connected to.
   ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" WITH BACKUP;
   ```

   Encrypt your database using tablespace encryption, column encryption, or both.

2.13. Rekeying or key rotation

After you have established your KeyControl Database Vault as the primary

protector for your master encryption keys, for security reasons you may want to periodically replace the keys, or re-key. For your particular system, you can do this by following the instructions below.

The following subsections show how to perform a re-key in Oracle multitenant environments. After re-key, the new encryption keys should be immediately available and usable by the client that initiated the re-key.

2.13.1. Rekey for a non-multitenant database

The `orcl.conf` file is used to utilize the access credentials for the KeyControl Database Vault, enabling the creation and utilization of the master key.

The following instructions begin by assuming the KeyControl (wallet) is already open.

CONNECT TESTER@DB, or CONNECT sysdba@DB

```
--Assumes KeyControl is already open
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf";
```

2.13.2. Rekey for a multitenant database with CDB and all the PDBs in one operation

CONNECT TESTER@CDB<n>

The following instructions begin by assuming the required CDB has started, and required PDBs and KeyControl (keystore) to be already open.

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" WITH BACKUP CONTAINER = ALL;
```

2.13.3. Rekey for a multitenant database with CDB only

The following instructions begin by assuming the required CDB has started and KeyControl (keystore) to be already open.

CONNECT TESTER@CDB<n>

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" WITH BACKUP;
```

2.13.4. Rekey for a multitenant database with a single PDB only

The `orcl.conf` file is used to utilize the access credentials for the KeyControl Database Vault, enabling the creation and utilization of the master key.

The following instructions begin by assuming the required CDB has started, the required PDB and KeyControl (keystore) to be already open.

CONNECT PDB<k>TESTER@CDB<n>PDB<k>

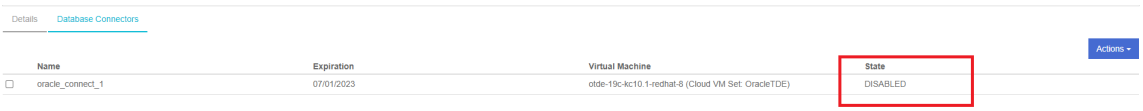
```
--Make the master key for the PDB you should be currently connected to
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" WITH BACKUP;
```

2.14. Enabling and Disabling Database Connector

To disable the Database Connector:

1. Log into the KeyControl Vault and navigate to **CLOUDKEYS** > **Key Sets**.
2. Select the desired Key Set and proceed to **Database Connectors**.
3. Choose the appropriate Database connector and access its settings.
4. Under **Actions**, locate the option to **Disable Connector**.

5. Select **Disable**.
6. When a confirmation message appears, select **Close**.
7. Confirm that the state is DISABLED.



8. Return to the Oracle Server in the SQL logged in as `sysdba`.
9. When you run the commands to verify the tables, you will notice that it shows the wallet is not open:

```
ERROR at line 1:
ORA-28365: wallet is not open
```

10. Confirm the wallet is closed with the following command:

```
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;
```

```
    CON_ID WRL_TYPE             STATUS
---------- -------------------- ------------------------------
         0 HSM                  CLOSED
```

To enable the Database Connector:

1. Log into the KeyControl Vault and navigate to **CLOUDKEYS** > **Key Sets**.
2. Select the desired Key Set and proceed to **Database Connectors**.
3. Choose the appropriate Database connector and access its settings.
4. Under **Actions**, locate the option to **Enable Connector**.

5. Open the keystore:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf";
```

6. Verify the opened keystore:

```
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;

    CON_ID WRL_TYPE             STATUS
---------- -------------------- ------------------------------
         0 HSM                  OPEN
```

Chapter 3. Troubleshooting

Oracle error messages may sometimes show error symptoms rather than the root cause. If you see an error you have not encountered before, search for further information online before attempting to resolve the error. If you remain unable to resolve the error, contact Oracle support.

If you edit an Oracle configuration file, use a simple text editor running on the host. Do not cut and paste the file contents from another file using a formatting editor, as it may insert hidden characters that are difficult to detect and which can stop the file from working. Entrust also suggests you avoid copying files onto a UNIX host via a Windows intermediary (this includes library files).

3.1. An SQL command is run, and there is no output, or an unexpected output or error occurs

1. Try reconnecting to the database.
2. If that does not work, try bouncing the database.

3.2. After a change to a configuration file, no resultant change in the database behavior is observed

1. Try reconnecting to the database.
2. If that does not work, try bouncing the database.

3.3. ORA-28367: wallet does not exist

1. Check that you have correctly installed and configured the Entrust PKCS#11 library.
2. Try reconnecting to the database.
3. Try bouncing the database.
4. Try restarting the Entrust hardserver.

3.4. ORA-28353: failed to open wallet

Check to see if your `/opt/oracle/entrust/orcl.conf` is formatted correctly. Ensure that the file is in a JSON format.

3.5. ORA-12162: TNS: net service name is incorrectly specified

Check that you have correctly set the value for ORACLE_SID in your local environment.

3.6. ORA-28407: Hardware Security Module failed with PKCS#11 error CKR_SESSION_HANDLE_INVALID(179)

It is necessary to update the PKCS#11 library for Oracle 19c and 21c Multitenant Database. For the most accurate and updated information regarding the PKCS11 library fix, nShield recommends referring to the KeyControl 10.1 release notes. This will provide comprehensive details on the issue and the fixed library.

3.7. ORA-03113: end-of-file on communication channel

It is necessary to update the PKCS#11 library for Oracle 19c and 21c Multitenant Database. For the most accurate and updated information regarding the PKCS11 library fix, nShield recommends referring to the KeyControl 10.1 release notes. This will provide comprehensive details on the issue and the fixed library.

Chapter 4. Additional resources and related products

4.1. Entrust digital security solutions

4.2. nShield product documentation