



Oracle Transparent Data Encryption and Entrust KeyControl Database Vault

Integration Guide

2025-02-07

Table of Contents

1. Introduction	1
1.1. Using this guide	1
1.2. Product configuration	2
1.3. Conventions used in this document	2
1.4. Overview	5
2. Procedures	7
2.1. Preparatory requirements	7
2.2. Create a Database Vault in the KeyControl Vault Server	7
2.3. Sign in to the Database Vault URL	11
2.4. Download the Policy Agent	11
2.5. Create a VMSet in KeyControl Vault	13
2.6. Create a key set	17
2.7. Link the P11 library	22
2.8. Test the integration	22
2.9. Open and close a keystore or KeyControl	23
2.10. Migrate from software keystore to KeyControl (multitenant)	23
2.11. Create master keys directly in KeyControl for a multitenant database	27
2.12. Rekeying or key rotation	31
2.13. Disable or enable the Database Connector	32
3. Troubleshooting	35
3.1. An SQL command is run, and there is no output, or an unexpected output or error occurs	35
3.2. After a change to a configuration file, no resultant change in the database behavior is observed	35
3.3. ORA-28367: wallet does not exist	35
3.4. ORA-28353: failed to open wallet	36
3.5. ORA-12162: TNS: net service name is incorrectly specified	36
4. Additional resources and related products	37
4.1. KeyControl	37
4.2. KeyControl as a Service	37
4.3. Entrust products	37
4.4. nShield product documentation	37

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, see the [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 guide shows support for multitenant databases. For more information on the multitenant support only by Oracle, see the [Oracle multitenant documentation](#).

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
KeyControl Vault	10.4.1

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

OS Version	Kernel	Oracle Version
Red Hat Enterprise Linux 9	Linux 5.14.0-503.21.1.el9_5.x86_64	Oracle Database 23ai Free Release - 23.0.0.0.0 - 23.6.0.24.10

1.3. Conventions used in this document

1.3.1. 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

- `<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
- `<database-identifier>`

This guide will use one following database identifies during a connection:

Oracle 23ai Free edition already deploys the following:

- `FREE` indicates the container database.
- `PDB1` indicates the pluggable database.

Multitenant database identifiers will be:

- `FREEEROT`, to connect to the `CDB$ROOT` for the container `FREE`.
- `FREEPDB1`, to connect to `PDB1` within `FREE`.

For example:

```
CONNECT sysdba@FREEEROT
CONNECT FREEPDB1TESTER@FREEPDB1
```

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:

- **CONNECT sysdba@FREEROOT** implies that if you are not already connected to **FREE**, then alter the session:

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

- **CONNECT FREEPDB1TESTER@FREEPDB1** implies that if you are not already connected to **FREEPDB1**, then alter the session:

```
ALTER SESSION SET CONTAINER = FREEPDB1;
```

Examples of **sqlplus** connection syntax for different users:

- sqlplus / as sysdba
- sqlplus / as sysdba@FREEROOT
- sqlplus FREEPDB1TESTER/Tester@//localhost:1521/FREEPDB1

1.3.2. 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, see [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

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 want to migrate encryption keys from an existing Oracle software keystore to Entrust KeyControl, or start directly with Entrust KeyControl.

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

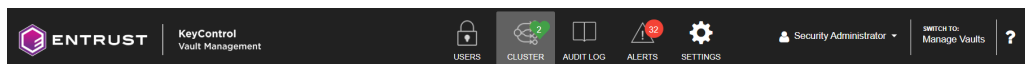
For more information on how to configure your Entrust environment, see the [KeyControl Vault Installation Guide](#).

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

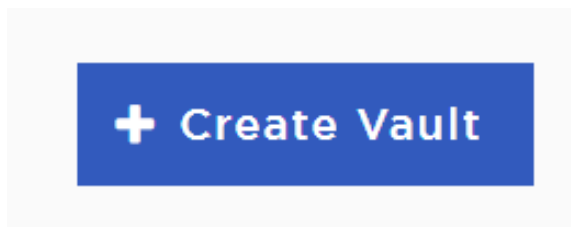
The KeyControl Vault appliance supports different type of vaults that can be used by all type of applications. This section describes how to create a Database Vault in the KeyControl Vault Server. See [Creating a Vault](#) for more details.

1. Log in to 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. If not in the **Vault Management** interface, in the top menu bar, on the right side, select **Switch to: Manage Vaults**.



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

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



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.

Vaults
Each vault has unique authentication and management

Create Vault
A vault will have unique authentication and management.

Type
Choose the type of vault to create

Database

Name *

Oracle-TDE

Description
Optionally add a short description to help identify this vault.

Vault for Oracle TDE

Max. 300 characters

Email Notifications OFF

⚠ SMTP needs to be configured to turn on email notifications

Use email to communicate with Vault Administrators, including their temporary passwords. Turning off email notifications means you will see and need to give temporary passwords to Vault Admins.

Administrator
Invite an individual to have complete access and control over this vault. They will be responsible for inviting additional members.

Admin Name *

Administrator

Admin Email *

xxxxx.xxx@xxxx.com

Create Vault Cancel

5. Select **Create Vault**.

✔ Vault Successfully Created

You will need to send the following information to the Vault Admin so they can log into their vault

Vault URL

[Blurred URL]

 Copy

User Name

[Blurred email address]

 Copy

Temporary Password

[Blurred password]

 Copy

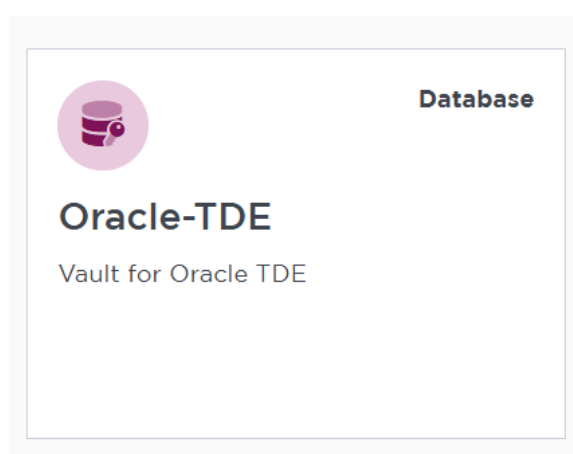
Close



A temporary password will be emailed to the administrator's email address. This is the password that will be used to sign in for the first time to the Database vault's space in KeyControl. In a closed-gap environment where email is not available, the password for the user is displayed when you first create the vault. That can be copied and sent to the user.

- 6. Select **Close** when the vault creation completes.

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



2.3. Sign in to the Database Vault URL

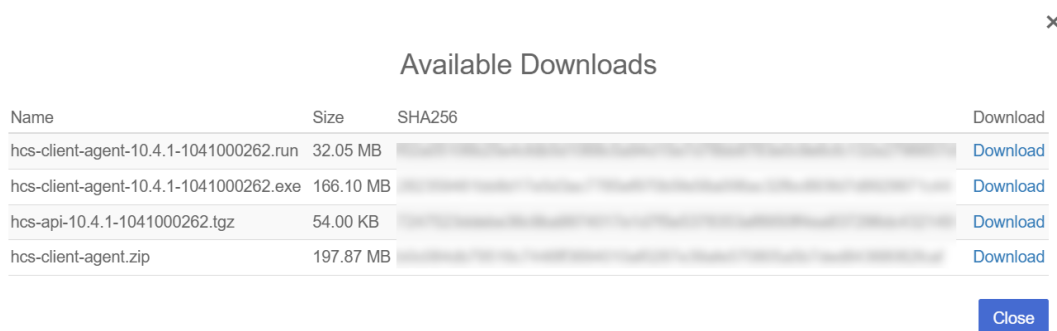
1. Sign in to the URL provided above with the temporary password that was copied.
2. Change the initial password when prompted.
3. Sign in again to verify.

2.4. Download the 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 in to the newly created vault:
 - a. Select the **WORKLOADS** tab.
 - b. Select **Actions > Download Policy Agent**

A list of available downloads appears. For example:



Name	Size	SHA256	Download
hcs-client-agent-10.4.1-1041000262.run	32.05 MB		Download
hcs-client-agent-10.4.1-1041000262.exe	166.10 MB		Download
hcs-api-10.4.1-1041000262.tgz	54.00 KB		Download
hcs-client-agent.zip	197.87 MB		Download

Close

2. Download the `hcs-client-agent-10.4.1-1041000262.run` file.

Once downloaded, transfer the file to the Oracle Server.

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
```

- 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:

```
% dnf install python3
```

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

```
% python --version
```

- Navigate to the directory where you downloaded the `hcs-client-agent-10.4.1-1041000262.run` file.

- Make the file executable:

```
% chmod +x hcs-client-agent-10.4.1-1041000262.run
```

- Run the installer:

```
% ./hcs-client-agent-10.4.1-1041000262.run
```

```
Verifying archive integrity... All good.
Uncompressing hcs-client-agent-10.4.1-1041000262.run 100%
x86_64
No Entrust Agent found on this system
Entrust Agent will be installed in /opt/hcs
Specify location for installing Entrust 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
```

- Verify the installation:

```
% 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)	
Other Devices			
Disk Name	Device Node	Status	
sda2	/dev/sda2	Mounted (/boot)	
sda1	/dev/sda1	Mounted (/boot/efi)	
sda3	/dev/sda3	LVM (rhel)	
rhel-swap	/dev/dm-1	Mounted (swap)	
rhel-home	/dev/dm-2	Mounted (/home)	
rhel-root	/dev/dm-0	Mounted (/)	

For more information, see [Entrust DataControl Policy Agent](#).

2.5. 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:

Create Cloud VM Set ✕

[VM Set](#) [Additional Properties](#) [Reauthentication Settings](#) [Key Encryption Key](#) [Single Encryption Key](#)

Name *

Group *

Description

Boundary Controls *

5. Select **Create**.

6. When a success message appears, click **Close**.

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

Actions - [VM Sets](#) [VMs](#) [Unauthenticated VMs](#) [Mappings](#) [Access Control Policies](#) [Active Directory](#) [Client Certificates](#) [Multi-Select: ☐](#) [Refresh ↻](#)

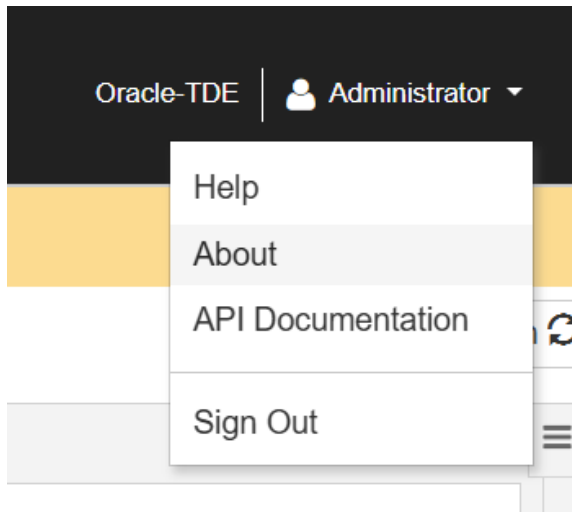
VM Set Name	Total V...	Group	Description
OracleTDE	0	Cloud Admin Group	OracleTDE Cloud VM Set

[Details](#) [Reauthentication Settings](#) [KeyIDs](#) [Asymmetric KeyIDs](#) [User Tasks](#) [System Tasks](#)

Name:	OracleTDE
Description:	OracleTDE Cloud VM Set
Group:	Cloud Admin Group
Total Registered VMs:	0
Boundary Controls:	Disabled
Heartbeat:	5 minutes
Grace Period:	1 days
Max Parallel Rekey Operations:	1
Rekey Interval:	0 days
Maximum VMs allowed:	Unlimited
Certificate Auto Renewal Period:	10 days
Certificate Expiration:	1 years
Single Encryption Key State:	Disabled <input type="button" value="Enable"/>
Auto Encryption:	Disabled
Decryption Allowed:	Yes
Policy Agent Uninstallation Allowed:	Yes

2.5.1. Register the Oracle Server VM to the VM set

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



2. Copy the Vault ID.



Vault Name: Oracle-TDE

Vault ID: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Version: KC 10.4.1 (b1041000262)

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:

```
Please provide the Vault login details
username: xxxxxx.xxxxxx@entrust.com
password:
```

```
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-23ai-kc1041 with KeyControl node(s) xx.xxx.xxx.xxx

Completing authentication for otde-23ai-kc1041 on KeyControl node(s) xx.xxx.xxx.xxx

Authentication complete, machine ready to use
Getting KeyControl Mapping information

KeyControl Mappings are not available

```

4. Verify the registration:

```
% hcl status
```

```

Summary
-----
KeyControl: xx.xxx.xxx.xxx:443
KeyControl list: xx.xxx.xxx.xxx:443
Vault ID: 63a0554c-afd3-40b1-832e-aa1398f82835
Status: Connected
Last heartbeat: Wed Jan 15 12:01:42 2025 (successful)
AES_NI: enabled
Certificate Expiration: Jan 15 17:00:10 2026 GMT
HTCRYPT: Not Installed

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

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

Other Devices
-----
Disk Name      Device Node      Status
-----
sda2           /dev/sda2       Mounted (/boot)
sda1           /dev/sda1       Mounted (/boot/efi)
sda3           /dev/sda3       LVM (rhel)
rhel-swap      /dev/dm-1       Mounted (swap)
rhel-home      /dev/dm-2       Mounted (/home)
rhel-root      /dev/dm-0       Mounted (/)

```

5. Enable TDE:

```
% hcl tde enable
```

```

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

If you are enabling TDE for an Oracle database, follow the steps mentioned below from the Administrator
Guide.
"Administration Guide > KeyControl Vault for Databases > KeyControl with Oracle TDE > Configuring the

```

Oracle Server Database"

6. Check the TDE status.

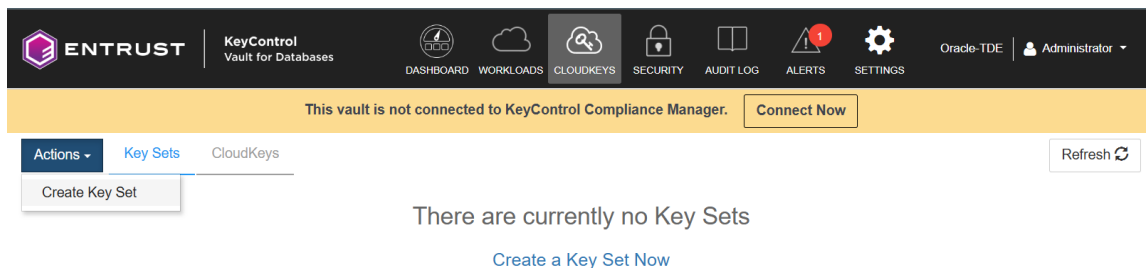
```
% hcl tde status
```

```
TDE is enabled on this VM
```

2.6. Create a key set

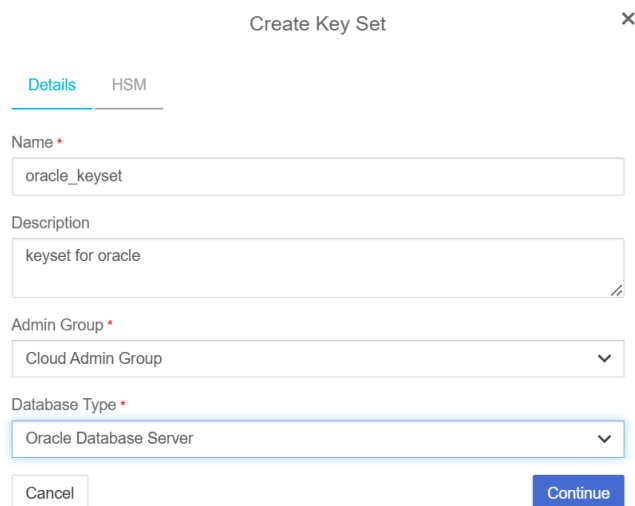
A KeySet in KeyControl Vault serves as a container for managing encryption keys used in various cryptographic operations.

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**.



The **Create Key Set** dialog appears.

4. In the **Details** tab, create the Key Set:
 - a. Enter a **Name**.
 - b. For **Admin Group**, select **Cloud Admin Group**.
 - c. For **Database Type**, select **Oracle Database Server**.



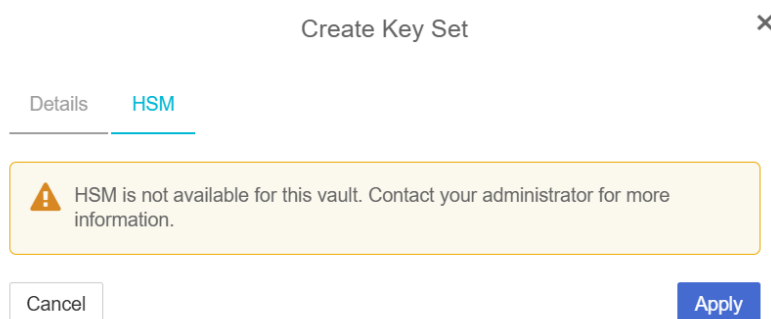
The screenshot shows a 'Create Key Set' dialog box with a close button (X) in the top right corner. It has two tabs: 'Details' (selected) and 'HSM'. The 'Details' tab contains the following fields:

- Name ***: A text input field containing 'oracle_keyset'.
- Description**: A text area containing 'keyset for oracle'.
- Admin Group ***: A dropdown menu with 'Cloud Admin Group' selected.
- Database Type ***: A dropdown menu with 'Oracle Database Server' selected.

At the bottom of the dialog, there are two buttons: 'Cancel' on the left and 'Continue' on the right.

5. Select **Continue**.

6. In the **HSM** tab, there is no HSM available, so just select **Apply**. 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.



The screenshot shows the 'Create Key Set' dialog box with the 'HSM' tab selected. A yellow warning box is displayed with the following message:

Warning: HSM is not available for this vault. Contact your administrator for more information.

At the bottom of the dialog, there are two buttons: 'Cancel' on the left and 'Apply' on the right.

For more information on HSM configuration with KeyControl, see [Hardware Security Modules with KeyControl Vault](#).

7. Select **Apply**.

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

8. Verify that your KeySet is listed.

2.6.1. Create the 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.
2. Select the **Database** tab.
3. Select **Create Connector Now**

The screenshot displays the Oracle KeyControl interface. At the top, there is a navigation bar with 'Actions' and 'Key Sets' tabs, and a 'Refresh' button. Below this is a table with the following columns: 'Key Set Name', 'Description', 'Admin Group', 'Database Type', and 'Keys'. The table contains one row with the following data: 'oracle_keyset', 'keyset for oracle', 'Cloud Admin Group', 'Oracle Database Server', and '0'. Below the table, there is a 'Details' section with a 'Database Connectors' tab. Under this tab, there is a table with columns 'Name', 'Expiration', 'Virtual Machine', and 'State'. The table is currently empty, and a message states 'There are currently no database connectors'. A 'Create Connector Now' button is visible. Below the table, there is a warning message: 'Before connecting to a database, you will need to enable TDE on the Oracle Database Server VM. Learn More'.

4. Create Database Connector:
 - a. Select the **VM Name**
 - b. Enter a **Connector Name**
 - c. Select an **Expiration**

✕

Create Database Connector

Create a connection to the database VM. You will need to make sure that the database has TDE enabled. [How do I enable TDE on the VM?](#)

Associate this connection with the following VM:

VM Name *
 ▼
[Don't see a VM to use?](#)

Connector Name *

Expiration *
 Never Choose a date

- 5. Select **Create**
- 6. Select the newly created database connector > **Actions** > **Generate Access Token**

Details Database Connectors			
Name	Expiration	Virtual Machine	State
<input checked="" type="checkbox"/> oracletde-connector	02/01/2026	otde-23ai-kc1041 (Cloud VM Set: OracleTDE)	ENABLED

Actions ▼

- Create Connector
- Generate Access Token
- Update Expiration
- Disable Connector
- Delete Connector

- 7. Select **Generate Token** and it will display the newly generated token.
- 8. Copy the access token (**Identity** and **Secret**).

x

Generate Access Token

Generate an access token and select Copy Config to save its content to be used in Oracle Server for configuring External Keystore and Master Encryption Key. [Learn More](#)

VM Name: **otde-23ai-kc1041 (Cloud VM Set: OracleTDE)**

Generate Token

Identity
Copy

oracletde-connector

Secret
Copy

Copy Config

Close

9. Select **Close**.

10. 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.

First create the `/opt/oracle/entrust` directory.

```
% mkdir -p /opt/oracle/entrust
```

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

```
{
  "identity": "oracletde-connector",
  "secret": "aBC.....XyZ="
}
```

11. Set the following ownership and permissions on the `/opt/oracle/entrust` directory:

- **Owner:** oracle
- **Group:** oinstall
- **Permissions:** 775

```
% chown oracle:oinstall /opt/oracle/entrust
% 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, see [KeyControl Vault Key Set for TDE](#).

2.7. Link the P11 library

You must now configure the Oracle PKCS #11 Library folder to use the 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

Make sure that `ORACLE_BASE` points to `/opt/oracle`.

```
#
# 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
```

2.8. Test the integration

When testing the integration, make sure you use the instructions that are appropriate to your installation. These tests are suitable for a multitenant database and software keystore.

For this integration, it is necessary to update the PKCS#11 library for Oracle 23ai 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
```

2.9. Open and close 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.

Multitenant considerations

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

To open/close a keystore for the container database only

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

To open/close a keystore for the container and all PDBs it holds

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

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

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

2.10. Migrate 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.

Repeat the following procedure for each software keystore from which you want to migrate. Each container database 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, 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.

In the Oracle Server log in to the SQL database as `sysdba`.

```
CONNECT sysdba@FREEROOT
```

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

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

2.10.2. Prepare for key migration by running an SQL script

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

Session altered.

SQL> ALTER PLUGGABLE DATABASE ALL CLOSE;

Pluggable database altered.

SQL> ALTER SYSTEM SET TDE_CONFIGURATION='KEYSTORE_CONFIGURATION=HSM|FILE' CONTAINER = ALL;

System altered.

SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> SHOW PARAMETER TDE_CONFIGURATION;

NAME                                TYPE                                VALUE
-----                                -                                -
tde_configuration                    string                              KEystore_CONFIGURATION=HSM|FILE
```

2.10.3. Create an auto login

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

Session altered.

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

Pluggable database altered.

-- Create Auto Login
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE '/opt/oracle/admin/FREE/keystore-
folder/tde' IDENTIFIED BY KeystorePassword1;

keystore altered.
```

2.10.4. Migrate from the keystore to KeyControl

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

Session altered.

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

keystore altered.
```

2.10.5. Disable the auto login

Here we only have to move keystore wallet file out of the way.

```
sudo -u oracle mv /opt/oracle/admin/FREE/keystore-folder/tde/cwallet.sso /opt/oracle/admin/FREE/keystore-folder
/tde/cwallet.sso.backup
```

2.10.6. Bounce the database

```
SQL> shutdown immediate;

Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup;

ORACLE instance started.

Total System Global Area 1603787624 bytes
Fixed Size 5421928 bytes
Variable Size 419430400 bytes
Database Buffers 1174405120 bytes
Redo Buffers 4530176 bytes
Database mounted.
```

Database opened.

2.10.7. Close all keystores

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

Session altered.

-- Open all the PDBs.
SQL> ALTER PLUGGABLE DATABASE ALL OPEN READ WRITE;

Pluggable database altered.

-- Close all keystores
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE CONTAINER=ALL;

--Show the Keystores are closed
SQL> ALTER SESSION SET CONTAINER = FREEPDB1;
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;
```

CON_ID	WRL_TYPE	STATUS
3	FILE	CLOSED
3	HSM	CLOSED

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;
```

CON_ID	WRL_TYPE	STATUS
1	FILE	CLOSED
1	HSM	CLOSED
2	FILE	CLOSED
2	HSM	CLOSED
3	FILE	CLOSED
3	HSM	CLOSED

2.10.8. Open the HSM protection wallet on all databases

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

-- Open DB
SQL> ALTER DATABASE OPEN;

-- Open all the PDBs.
SQL> ALTER PLUGGABLE DATABASE ALL OPEN READ WRITE;

-- Open keystore
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" CONTAINER=ALL;

-- Show HSM Keystore as Open
SQL> ALTER SESSION SET CONTAINER = FREEPDB1;
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;
```

CON_ID	WRL_TYPE	STATUS
3	FILE	CLOSED

```
3 HSM OPEN
```

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;  
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;
```

CON_ID	WRL_TYPE	STATUS
1	FILE	CLOSED
1	HSM	OPEN
2	FILE	CLOSED
2	HSM	OPEN
3	FILE	CLOSED
3	HSM	OPEN

2.11. Create master keys directly in KeyControl for a multitenant database

The following procedure applies when the target database is multitenant, and there is no preexisting software keystore.

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, 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 database master key first. After the container database master key has been created, you have a choice of how you create master keys for all PDBs:

- in one operation
- for each PDB individually



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

In the Oracle Server log in to the SQL database as **sysdba**.

```
CONNECT sysdba@FREEROOT
```

2.11.1. Set the WALLET_ROOT and TDE_CONFIGURATION parameters

To set and use the WALLET_ROOT and TDE_CONFIGURATION parameters:

1. Set up the **WALLET_ROOT** parameter.

You must set up the **WALLET_ROOT** parameter even if you do not use a keystore. The database needs to be bounced after setting up the **WALLET_ROOT** parameter.

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

SQL> shutdown immediate;

SQL> startup

SQL> SHOW PARAMETER WALLET_ROOT;
```

NAME	TYPE	VALUE
wallet_root	string	/opt/oracle/entrust

2. Set up the **TDE_CONFIGURATION** parameter.

The database needs to be bounced after setting up the **TDE_CONFIGURATION** parameter.

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

Session altered.

SQL> ALTER PLUGGABLE DATABASE ALL CLOSE;

Pluggable database altered.

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

System altered.

SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> SHOW PARAMETER TDE_CONFIGURATION;
```

NAME	TYPE	VALUE
tde_configuration	string	KEYSTORE_CONFIGURATION=HSM

```
SQL> shutdown immediate;

SQL> startup
```

2.11.2. Configure Oracle to generate the master encryption key

```
SQL> GRANT ADMINISTER KEY MANAGEMENT TO SYSTEM;

Grant succeeded.

SQL> GRANT RESOURCE TO syskm;
```

```

Grant succeeded.

SQL> GRANT UNLIMITED TABLESPACE TO syskm;

Grant succeeded.

SQL> commit;

Commit complete.

```

2.11.3. Open the HSM keystore

You will see that when you open the HSM Keystore, the status will say **OPEN_NO_MASTER_KEY**.

```

SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

-- Open DB
SQL> ALTER DATABASE OPEN;

-- Open all the PDBs.
SQL> ALTER PLUGGABLE DATABASE ALL OPEN READ WRITE;

-- Open keystore
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "file:/opt/oracle/entrust/orcl.conf" CONTAINER=
ALL;

-- Show HSM Keystore as open
SQL> ALTER SESSION SET CONTAINER = FREEPDB1;

Session altered.

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

   CON_ID WRL_TYPE          STATUS
-----
        3 HSM                OPEN_NO_MASTER_KEY

SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

Session altered.

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

   CON_ID WRL_TYPE          STATUS
-----
        1 HSM                OPEN_NO_MASTER_KEY
        2 HSM                OPEN_NO_MASTER_KEY
        3 HSM                OPEN_NO_MASTER_KEY

```

2.11.4. Create a TDE master encryption key

The TDE Master Encryption Key is stored inside the Entrust KeyControl. Oracle Database uses the TDE master encryption key to encrypt or decrypt TDE table keys and tablespace keys.

```
CONNECT sysdba@FREEROOT
```

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

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

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

keystore altered.

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

The master key is created.

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

2.11.5. Look for the newly created cloud key in the KeyControl Database Vault

1. Go to **CLOUDKEYS** in the top bar.
2. Select the **CloudKeys** Tab.
3. Select the KeySet.

You will find the newly created cloud key.

The screenshot shows the KeyControl interface. At the top, there's a navigation bar with 'ENTRUST KeyControl Vault for Databases' and various menu items like DASHBOARD, WORKLOADS, CLOUDKEYS, SECURITY, AUDIT LOG, ALERTS, and SETTINGS. A notification banner states 'This vault is not connected to KeyControl Compliance Manager. Connect Now'. Below this, there are tabs for 'Key Sets' and 'CloudKeys'. A dropdown menu shows 'oracle_keyset (TDE)'. The main area displays a table of CloudKeys:

CloudKey Name	Description	Expires	Key Status
5462c711-475-429-4121-464889323	DATA_OBJECT_SUPPORTED_IDEN	Never	AVAILABLE
5717376-284-464-448-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE
464444-42718444275-42718444275	ORACLE_TDE_464444-42718444275	Never	AVAILABLE

Below the table, there are tabs for 'Details', 'Tags', and 'Versions'. The 'Details' tab is active, showing the following information:

- Name: 5462c711-475-429-4121-464889323
- Description: DATA_OBJECT_SUPPORTED_IDEN
- Key Status: AVAILABLE
- Key Version: 464444-42718444275-42718444275
- Key Source: KEYCONTROL
- Key Set: oracle_keyset
- Expires: Never Choose a date
- Cipher: Object
- Key Type: OBJECT

2.12. 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.

In the Oracle Server log in to the SQL database as **sysdba**.

```
CONNECT sysdba@FREEROOT
```

2.12.1. Rekey for a multitenant database

1. Doing it for CDB and all the PDBs in one operation.

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

Session altered.

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

```
Pluggable database altered.

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

keystore altered.
```

2. Doing for the CDB only.

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

2.13. Disable or enable the Database Connector

2.13.1. Disable the Database Connector

1. Log in to the KeyControl Database Vault.
2. Select **CLOUDKEYS** in the top bar.
3. Select the **Key Sets** tab.
4. Select the desired Key Set and proceed to **Database Connectors**.
5. Choose the appropriate Database connector and access its settings.
6. Under **Actions**, locate the option to **Disable Connector**.

The screenshot shows the Entrust KeyControl Database Vault interface. At the top, there is a navigation bar with 'ENTRUST KeyControl Vault for Databases' and several menu items: DASHBOARD, WORKLOADS, CLOUDKEYS (selected), SECURITY, AUDIT LOG, ALERTS (with a notification badge), and SETTINGS. Below the navigation bar, a yellow banner states 'This vault is not connected to KeyControl Compliance Manager.' with a 'Connect Now' button. The main content area is titled 'Key Sets' and contains a table with the following data:

Key Set Name	Description	Admin Group	Database Type	Keys
oracle_keyset	keyset for oracle	Cloud Admin Group	Oracle Database Server	29

Below the table, there are tabs for 'Details' and 'Database Connectors' (selected). The 'Database Connectors' section shows a table with the following data:

Name	Expiration	Virtual Machine	State	Actions
<input checked="" type="checkbox"/> oracleTde-connector	02/01/2026	otde-23ai-kc1041 (Cloud VM Set: OracleTDE)	ENABLED	<ul style="list-style-type: none"> Create Connector Generate Access Token Update Expiration Disable Connector Delete Connector

7. Select **Disable**.
8. Confirm that the state is DISABLED.

Details		Database Connectors		Actions	
Name	Expiration	Virtual Machine	State		
<input type="checkbox"/> oracletdc-connector	02/01/2026	otde-23ai-kc1041 (Cloud VM Set: OracleTDE)	DISABLED		

9. Return to the Oracle Server in the SQL logged in as **sysdba**.
10. 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
```

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

```
SQL> ALTER SESSION SET CONTAINER = FREEPDB1;
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;

CON_ID WRL_TYPE          STATUS
-----
      3 HSM                CLOSED

SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;

CON_ID WRL_TYPE          STATUS
-----
      1 HSM                CLOSED
      2 HSM                CLOSED
      3 HSM                CLOSED
```

2.13.2. Enable the Database Connector

1. Log in to the KeyControl Database Vault.
2. Select **CLOUDKEYS** in the top bar.
3. Select the **Key Sets** tab.
4. Select the desired Key Set and proceed to **Database Connectors**.
5. Choose the appropriate Database connector and access its settings.
6. Under **Actions**, locate the option to **Enable Connector**.

Details Database Connectors				Actions
Name	Expiration	Virtual Machine	State	
<input checked="" type="checkbox"/> oracletde-connector	02/01/2026	otde-23ai-kc1041 (Cloud VM Set: OracleTDE)	DISABLED	<ul style="list-style-type: none"> Create Connector Generate Access Token Update Expiration Delete Connector Enable Connector

7. Open the keystore:

Return to the Oracle Server in the SQL logged in as **sysdba**.

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;
```

-- Open DB

```
SQL> ALTER DATABASE OPEN;
```

-- Open all the PDBs.

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN READ WRITE;
```

-- Open keystore

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

-- Show HSM Keystore as open

```
SQL> ALTER SESSION SET CONTAINER = FREEPDB1;
```

Session altered.

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

CON_ID	WRL_TYPE	STATUS
3	HSM	OPEN

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;
```

Session altered.

```
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

2.13.3. Check that you can see the encrypted table content in plaintext

You should be able to do queries on the encrypted tables and see the table content in plaintext.

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 hardware server.

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.

Chapter 4. Additional resources and related products

[4.1. KeyControl](#)

[4.2. KeyControl as a Service](#)

[4.3. Entrust products](#)

[4.4. nShield product documentation](#)