Oracle Transparent Data Encryption with Data Guard and Entrust Cryptographic Security Platform Key Management Vault

Integration Guide

2026-04-22

© Entrust Corporation. All rights reserved.

Table of Contents

1. Introduction . . . 1
    1.1. Using this guide . . . 1
    1.2. Product configuration . . . 3
    1.3. Conventions used in this document . . . 3
    1.4. Overview . . . 5
2. Procedures . . . 7
    2.1. Preparatory requirements . . . 7
    2.2. Create a Database Vault in the Key Management Vault Server . . . 8
    2.3. Sign in to the Database Vault URL . . . 10
    2.4. Create a Node Mapping . . . 10
    2.5. Download TDE script bundle . . . 11
    2.6. Entrust Cryptographic Security Platform Key Management Vault client setup on first Oracle database server node (primary) . . . 14
    2.7. Entrust Cryptographic Security Platform Key Management Vault client setup on second Oracle database server node (standby) . . . 17
    2.8. Enable TDE on non-encrypted Oracle Database . . . 18
    2.9. Migrate from software wallet to Key Management Vault . . . 26
    2.10. Rotation of the TDE Key . . . 28
    2.11. Disable or enable the Database Connector . . . 30
    2.12. Reverse Migration from Key Management Vault to software wallet . . . 32
    2.13. Backup or restore . . . 33
3. Troubleshooting . . . 34
    3.1. An SQL command is run, and there is no output, or an unexpected output or error occurs . . . 34
    3.2. After a change to a configuration file, no resultant change in the database behavior is observed . . . 34
    3.3. ORA-28367: wallet does not exist . . . 34
    3.4. ORA-28353: failed to open wallet . . . 34
    3.5. ORA-12162: TNS: net service name is incorrectly specified . . . 35
4. Example command output . . . 36
    4.1. encrypt.sh . . . 36
    4.2. setup.sh . . . 46
5. Additional resources and related products . . . 48
    5.1. Entrust CSP Key Manager . . . 48
    5.2. Entrust products . . . 48
    5.3. nShield product documentation . . . 48

Chapter 1. Introduction

This guide describes the integration of the Entrust Cryptographic Security Platform Key Management Vault with an Oracle database using Oracle Data Guard. Oracle Data Guard offers a collection of services that create, maintain, manage, and monitor one or more standby databases to enable Oracle databases to survive disasters and data corruptions. It ensures high availability, data protection, and disaster recovery for enterprise data. Transparent Data Encryption (TDE) complements Oracle Data Guard by encrypting sensitive data stored in the databases. It safeguards against unauthorized access when the data is at rest, adding an additional security layer. TDE automatically encrypts and decrypts data at the storage level, thus preserving the usability and performance of the Oracle databases while seamlessly integrating with Oracle Data Guard to ensure that both primary and standby databases adhere to the same security standards without manual intervention. Entrust Cryptographic Security Platform Key Management Vault acts as an Extensible Key Management (EKM) solution for Oracle that securely manages keys and encrypts sensitive data using Transparent Data Encryption (TDE).

- For more detailed information on Entrust Cryptographic Security Platform Key Management Vault Database Vaults, see Cryptographic Security Platform Vault for Databases Online 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 and column encryption.

This guide focus on how TDE is used and what needs to happen at the Oracle Data Guard configuration to ensure high availability, data protection, and disaster recovery for enterprise data that has been encrypted using Entrust Cryptographic Security Platform Key Management Vault Database Vault.

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 Cryptographic Security Platform Key Management Vault 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 Entrust Key Management Vault protection.
    - Begin using Entrust Key Management Vault protection immediately if no Oracle software wallet or keystore already exists.
- Examples and advice on how the product may be used.
- Oracle Data Guard configuration with Entrust Key Management Vault especially after encryption has been put in place.
- Troubleshooting advice.

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

Assuming you already have your Oracle database installed and Oracle Data Guard properly configured and working, after installing and configuring the Entrust Cryptographic Security Platform Key Management Vault 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 Cryptographic Security Platform Key Management Vault Database Vault with an Oracle database with Data Guard in place, 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 Key Management Vault Database Vaults and have a basic understanding of the setup processes involved in configuring Oracle database TDE and Oracle Data Guard. Familiarity with these concepts will ensure a smoother implementation of the integration.

It is important to note that this guide uses a single node primary and standby Oracle servers and a RAC environment was not used for testing. Taking this into account, some SQL statements used in the guide may vary slightly when used in a RAC setup.

1.2. Product configuration

Entrust has successfully tested Entrust Cryptographic Security Platform Key Management Vault Database Vault with the following configurations:

Vendor	Product	Version
VMware	vSphere	8.0
Entrust	Cryptographic Security Platform	1.3
Entrust	Key Management Vault	10.5.1
Oracle	Oracle Database Enterprise Edition	19C - 19.3.0.0.0
Red Hat	Red Hat Enterprise	Linux 8

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.

1. `<database-user>`

   This guide will use one following users for connecting to databases:

   - `sysdba`, Oracle's standard `sysdba` user
   - `tester`, User created to perform table column encryption.

Examples of `sqlplus` connection syntax:

```
% sqlplus / as sysdba
```

To connect as different users:

```
CONNECT tester/<test_user_password>;
CONNECT sys/<sys_user_password> as sysdba;
```

2. `<database-identifier>`

   This guide will use one following database identifiers for connecting to databases:

   - `CDB1ROOT`, to connect to the `CDB$ROOT` for the container `CDB1`.
   - `CDB1PDB1`, to connect to `PDB1` within `CDB1`.

1.3.2. Key migration and legacy keys

Entrust Key Management Vault 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 Entrust Key Management Vault is a software-based solution and not a physical hardware security module (HSM).

Entrust Key Management Vault 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.

Entrust Key Management Vault offers support for various HSMs, including nShield, Luna, and cloud HSMs.

For more information on HSM configuration with KeyControl, see Hardware Security Modules with Cryptographic Security Platform Vault.

Encryption master keys can be migrated between an existing Oracle keystore and Entrust Key Management Vault 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 the Entrust Key Management Vault wallet. Instead, fresh master key(s) are created within the software keystore or Entrust Key Management Vault 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 the Entrust Key Management Vault wallet where they were originally created. After performing a key migration, you can retain access to the legacy keys in the software keystore or Key Management Vault 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 Key Management Vault 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 Key Management Vault 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 Entrust Cryptographic Security Platform Key Management Vault for managing Transparent Data Encryption (TDE) within the Oracle environment. Firstly, it increases visibility into TDE keys, providing administrators with better oversight and control. Entrust Key Management Vault 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, Entrust Key Management Vault 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 Entrust Key Management Vault, enhancing overall security. Encryptions keys are securely stored on a FIPS 140 Level 1 certified Encrypted Object store, ensuring compliance with stringent security standards.

Entrust Cryptographic Security Platform Key Management Vault 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 Entrust Key Management Vault, providing a comprehensive record of key management activities for compliance and auditing purposes. Overall, leveraging Key Management Vault 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 Cryptographic Security Platform Key Management Vault Database 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. Oracle Data Guard is setup and working with a minimum of one standby server in place.

To integrate an Oracle database with Entrust Cryptographic Security Platform Key Management Vault Database Vault and Oracle Data Guard, the following steps are required:

1. Configure the environment.
2. Install the Entrust Cryptographic Security Platform Key Management Vault Database Vault software.
3. Configure Oracle database software to use the Entrust Cryptographic Security Platform Key Management Vault Database Vault.
4. Modify Oracle Data Guard configuration so encrypted data on primary server is also migrated and visible on the standby servers.

Details of your installation and configuration will depend on whether you want to migrate encryption keys from an existing Oracle software keystore to Entrust Cryptographic Security Platform Key Management Vault, or start directly with Entrust Cryptographic Security Platform Key Management Vault.

The default host server user is `oracle` unless stated otherwise. The example database used in this guide is `CDB1` unless stated otherwise.

For more information on how to configure your Entrust environment, see Key Management Vault Installation and Upgrade Guide.

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

2.1.1. Oracle Data Guard Environment

For the purpose of this integration procedure, we will use a primary and standby server for Oracle Data Guard. For reference the following table has been created so the user can visualize the environment:

Site	Primary	Standby
DB_NAME	CDB1	CDB1
INSTANCE_NAME	CDB1	CDB1
DB_UNIQUE_NAME	CDB1P	CDB1S
HOST_NAME	db1	db2

2.1.2. Oracle Environment Variables

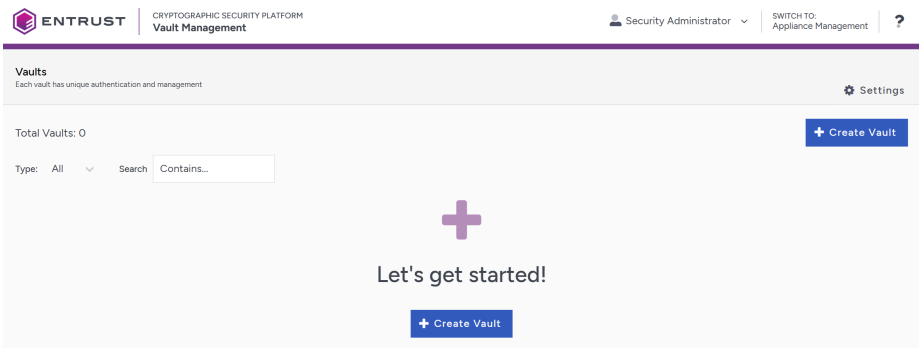
The following environment variables were used throughout the guide for Oracle:

```
export ORACLE_BASE=/opt/oracle
export ORACLE_HOME=$ORACLE_BASE/product/19c/dbhome_1
export ORACLE_SID=CDB1
export DATABASE_NAME=CDB1
```

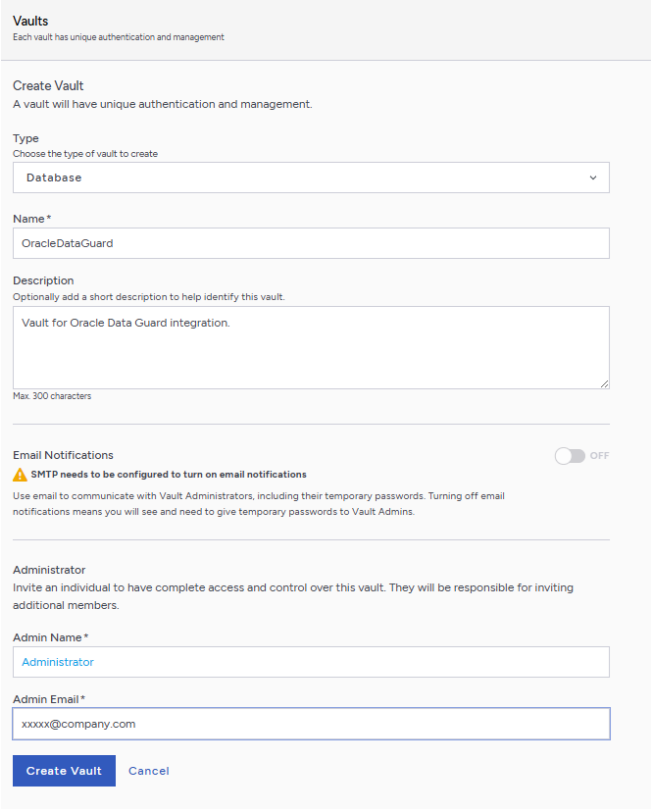
2.2. Create a Database Vault in the Key Management Vault Server

The Key Management 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 Key Management Vault Server. Refer to Creating a Vault for more details.

1. Log in to the Key Management Vault server in your web browser using the **secroot** credentials to access the IP address of the server.
2. If you are not in the vault Management interface, select **SWITCH TO: Manage Vaults** in the Menu Header
3. Select **Create Vault**.

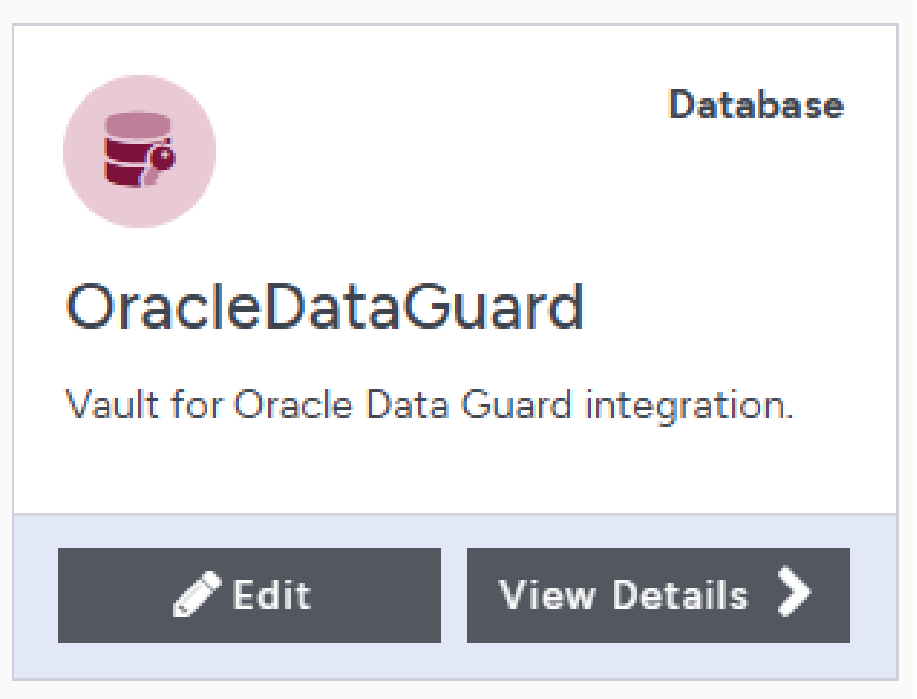


4. Create a **Database** Vault:
    - For **Type**, select **Database**.
    - For **Name**, enter the name of the vault.
    - For **Description**, enter the description of the vault.
    - For **Admin Name**, enter the name of the administrator of the vault.
    - For **Admin Email**, enter a valid 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 sign in for the first time to the Database Vault space in the Key Management Vault. 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.

5. Select **Create Vault**.
6. Select **Close** when the vault creation completes.
7. The newly vault is added to the vault dashboard.



2.3. Sign in to the Database Vault URL

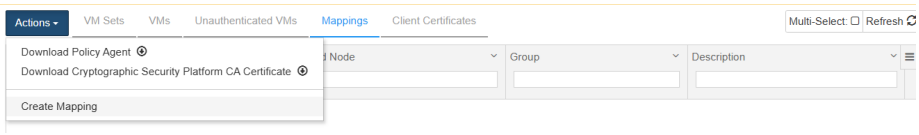
1. Sign in to the Database Vault URL provided when it was created with the temporary password that was copied.
2. Change the initial password when prompted.
3. Sign in again to verify.

2.4. Create a Node Mapping

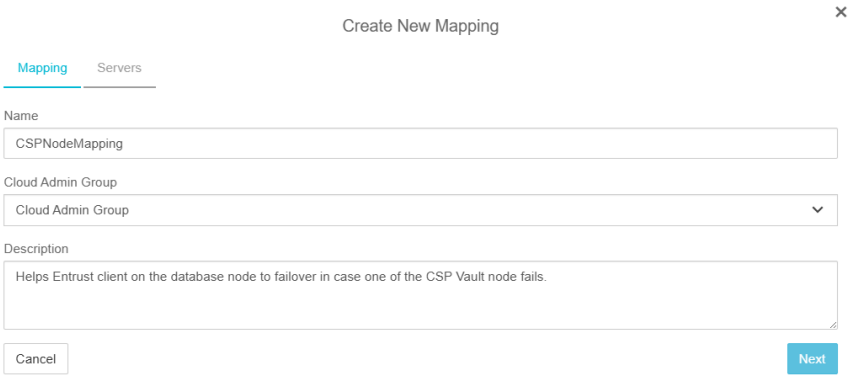
A Key Management Vault node mapping helps Entrust clients on the database server nodes to failover in case one of the Key Management Vault node fails.

In the Cryptographic Security Platform Key Management Vault Database Vault web GUI:

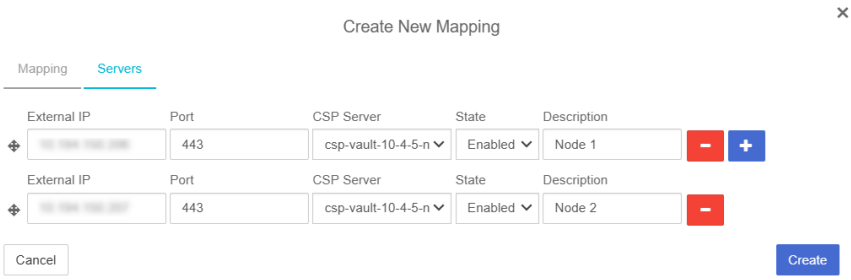
1. Navigate to **Workloads**
2. Select the **Mappings** tab.
3. In the **Actions** menu choose **Create Mapping** to create a new mapping.



4. In the **Create New Mapping** Dialog, in the **Mapping** tab, enter the following:
    - **Name**: Name of the mapping.
    - **Cloud Admin Group**: Select **Cloud Admin Group** (default).
    - **Description**: Description of the mapping.
    - Select **Next**.



5. In the **Create New Mapping** Dialog, in the **Servers** tab, enter the following for each Node in the Key Management Vault cluster:
    - **External IP**: IP address of the node.
    - **Port**: Enter **443**.
    - **CSP Server**: Select the Key Management Vault Server.
    - **State**: Select **Enabled**.
    - **Description**: Description of the node.



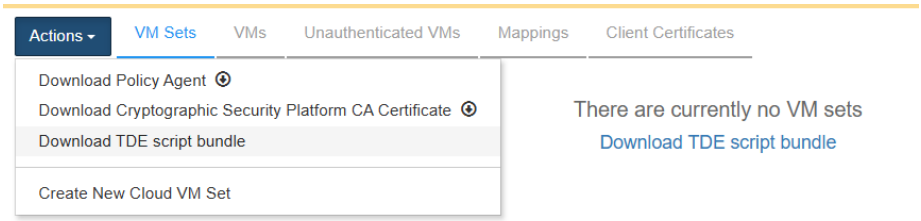
6. Select **Create**.

    Once created the mapping gets listed.

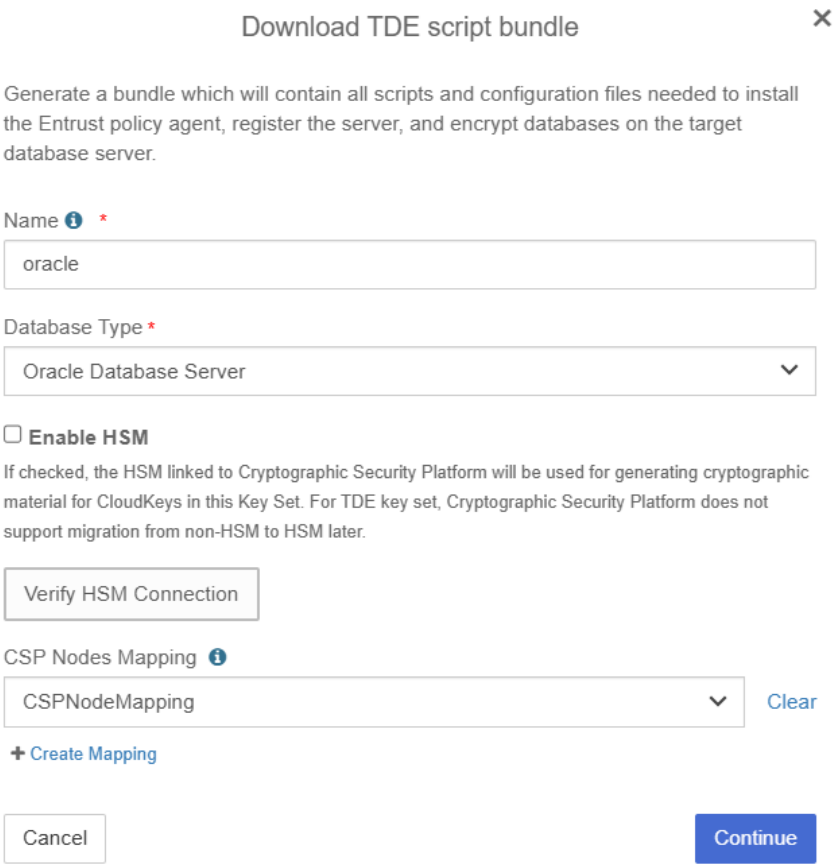
2.5. Download TDE script bundle

In the Cryptographic Security Platform Key Management Vault Database Vault web GUI:

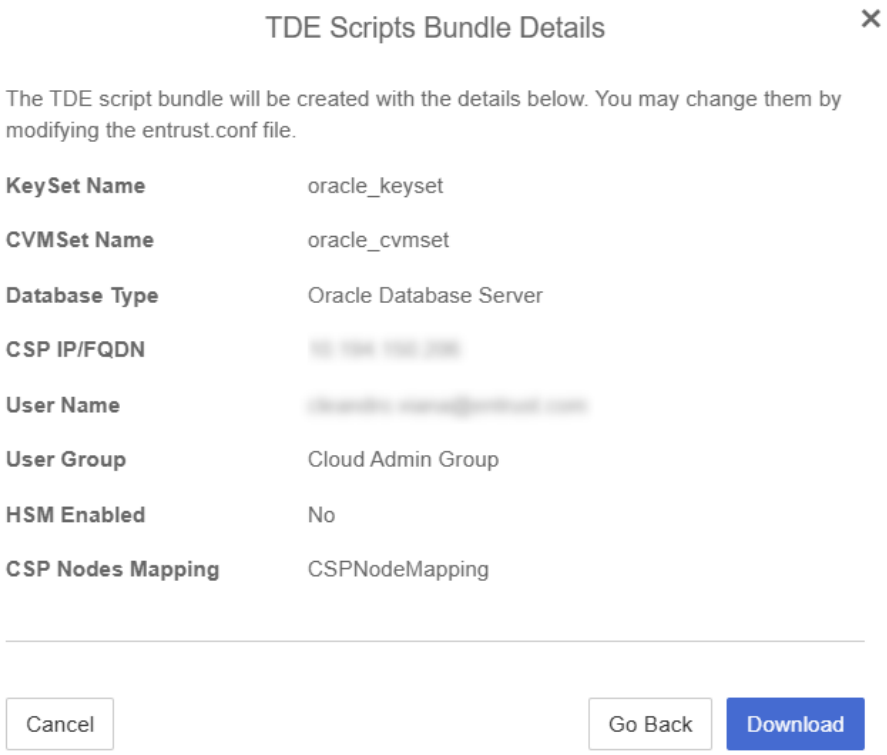
1. Navigate to **Workloads**.
2. Select the **VM Sets** tab.
3. In the **Actions** menu select **Download TDE script bundle**.



4. In the **Download TDE Scripts bundle** Dialog, enter the following:
   - **Name**: Provide a Name which will become prefix for your Keyset and CVMset. For example, if you choose **oracle** as Name then the setup script will create **oracle_keyset** and **oracle_cvmset** respectively.
   - **Database Type**: Select **Oracle Database Server**.
   - **Enable HSM**: If have an HSM configured and want to protect your TDE master keys with it, then choose **Enable HSM**. You will be required to verify the HSM connection to proceed, if you enable HSM.
   - **CSP Nodes Mapping**: In the drop down choose the **CSP Nodes Mapping**, if you have created one, or proceed without one.
   - Select **Continue**.



5. The **TDE Scripts Bundle Details** appear. Validate the information is correct and Select **Download**.



A zip file named with the name given in the **Download TDE Scripts bundle** dialog is downloaded, that is, `oracle.zip`. Save this file as this will be used in each Oracle database instance server.

2.6. Entrust Cryptographic Security Platform Key Management Vault client setup on first Oracle database server node (primary)

1. Log in to the primary Oracle server as the `oracle` user.
2. Create a directory for the bundle in the oracle's user home directory.

```
% mkdir ~/csp
```

3. Copy the TDE bundle downloaded earlier to the Primary Oracle server node.

   The bundle is a zip file <name>.zip with <name> as provided by the admin while downloading the bundle. For this guide the name used was `oracle`, so the name of the file is `oracle.zip`.

```
% scp oracle.zip oracle@xx.xxx.xxx.xxx:/home/oracle/csp/.
```

4. Extract the files in the bundle:

```
% cd ~/csp; unzip oracle.zip

Archive:  oracle.zip
  inflating: oracle/get_token.sh
  inflating: oracle/encrypt.sh
  inflating: oracle/setup.sh
  inflating: oracle/dgmappings.py
  inflating: oracle/entrust.conf
```

5. Modify `entrust.conf` file to match the environment.

   `entrust.conf` is a config file which has most of the parameters pre-filled. You can still modify these values if you think that is required. There are some parameters which are specific to the Oracle database server and they need to be modified. Here is a list of the parameters which need to be set by the DB administrator:

   oracle_user

   this is the name of the database admin user, typically the value is `oracle`

oracle_user_group

The group to which **oracle_user** belongs, typically the value is `oinstall`.

Here is the `entrust.conf` for our environment:

```
#
# kind of a cluster or setup name for your oracle RAC
# it should be unique in a vault
# Make sure these same parameters are used from all the nodes in your RAC cluster and DR nodes
#
dbset_name="oracle"

#
# KeyControl parameters
# ---------------------
#
kcv_ip_or_fqdn="xx.xxx.xxx.xxx"

#
# If KeyControl mapping has been setup on the KC cluster, it can be provided here
#
kc_mapping="CSPNodeMapping"

#
# If KeyControl is configured with HSM then it can be enabled using "yes" as value
# if not specified here then the default value is yes if KC has hsm enabled
#
enable_hsm="no"

#
# Vault parameters
# ----------------------
vaultid="4b551f9d-0442-45e1-ad6e-6a45ec2ce0af"
user_name="xxxx.xxxx@entrust.com"

#
# Uncomment the password parameter, if you want to set it here
# if you do not provide password for vault admin then setup script will prompt you
#
# password="password23!"
#
user_group="Cloud Admin Group"

#
# Oracle user parameters
# ----------------------
#
oracle_user="oracle"
oracle_user_group="oinstall"

#
# hostname suffix is used only in qa testing, please do not uncomment it
# hostname_suffix="dr"
```

6. After editing entrust.conf, save the changes.

2.6.1. Set up the Client

1. Run the `setup.sh` script as the `root` user or using `sudo`.

```
% sudo ./setup.sh first entrust.conf
```

We are doing this on the primary server, so we use `first` as the first argument.

When we use this option, the `keyset`/`vmset` are created.

The script asks for the Key Management Vault Database vault administrator password and it prints the name of the access token file:

```
Saving access token in file `/opt/oracle/entrust/oracle.conf`.
```

Make a note of this as it will be used in the subsequent SQL statements to open the wallet.

For an example of the full output for this command, see here.

If the HSM is enabled in the settings of the Key Management Vault appliance manager and in the `entrust.conf` file **enable_hsm** is set to **no**, you may get a message like this when you run the `setup.sh` script:

```
Create keyset oracle_keyset
Server returned error: Failed to create Keyset root key

Failed to download policy agent
```

If that is the case, disable the HSM settings in Key Management Vault to resolve the issue. If you selected to use the HSM (**enable_hsm** set to **yes**), then the HSM settings in the Key Management Vault must be enabled.

2. Set up the environment for the `encrypt.sh` script:

```
% ./encrypt.sh setenv
```

For an example of the full output for this command, see here.

3. Check the status:

```
% ./encrypt.sh status
```

For an example of the full output for this command, see here.

2.7. Entrust Cryptographic Security Platform Key Management Vault client setup on second Oracle database server node (standby)

1. Login to the standby server as the `oracle` user.
2. Create a directory for the bundle in the oracle's user home directory.

```
% mkdir ~/csp
```

3. Copy the TDE bundle downloaded earlier to the standby Oracle server node.

```
% scp oracle.zip oracle@xx.xxx.xxx.xxx:/home/oracle/csp/.
```

4. Extract the files in the bundle:

```
% cd ~/csp; unzip oracle.zip

Archive:  oracle.zip
  inflating: oracle/get_token.sh
  inflating: oracle/encrypt.sh
  inflating: oracle/setup.sh
  inflating: oracle/dgmappings.py
  inflating: oracle/entrust.conf
```

5. Copy the modified entrust.conf from the first (primary) Oracle server node and overwrite the config file from the bundle.

```
% scp oracle@xx.xxx.xxx.xxx:/home/oracle/csp/entrust.conf .
```

6. Run `setup.sh` as the `root` user or using `sudo`.

```
sudo ./setup.sh other entrust.conf
```

   The value of first argument is `other`.

   For an example of the full output for this command, see here.

7. Set the environment for the standby setup.

```
% ./encrypt.sh setenv
```

   For an example of the full output for this command, see here.

8. Perform initial setup of the standby server.

```
% ./encrypt.sh standby setup
```

For an example of the full output for this command, see here.

9. Check the status

```
% ./encrypt.sh status
```

For an example of the full output for this command, see here.

2.8. Enable TDE on non-encrypted Oracle Database

This section describes how to enable TDE if the database has not been previously encrypted. If your database is already encrypted using a software wallet, go to section Migrate from software wallet to Key Management Vault.

The setup in this section uses a primary server and a standby server. No RAC is used. This setup uses CSP Vault as the keystore.

Use the `encrypt.sh` script included in the bundle to enable database encryption. This script must be run as the `oracle` user.

We will do this on the primary server.

1. Set the environment.

   The Oracle administrator must provide the following parameters to set the context. The following operation creates a file named `<dbset_name>.env`, for example `oracle.env`, in the script directory.

   ```
   % ./encrypt.sh setenv
   ```

   For an example of the full output for this command, see here.

2. Check the status.

   After setting the environment variables, the administrator should check the status to determine the current state of the Oracle database server.

   ```
   % ./encrypt.sh status
   ```

   For an example of the full output for this command, see here.

3. Encrypt the database:

```
% ./encrypt.sh encrypt
```

   For an example of the full output for this command, see here.

4. Check the status.

```
% ./encrypt.sh status
```

   For an example of the full output for this command, see here.

2.8.1. Database Encryption with TDE keys

There are two types of encryptions considered in the guide: column level encryption and tablespace level encryption.

Connect to the database as SYSDBA.

```
% sqlplus / as sysdba
```

2.8.1.1. Column encryption

Since Columns are SYS owned objects, you will run into this error when attempting to encrypt a column in a database:

```
ERROR at line 1:
ORA-28336: cannot encrypt SYS owned objects
```

To get around the problem you need to create a user and grant the necessary permissions. You will use this user to create the table and encrypt the column.

1. Create the User

```
SQL> CREATE USER TESTER IDENTIFIED BY test_password;

User created.
```

2. Grant necessary privileges

```
SQL> GRANT CREATE SESSION, CREATE TABLE TO TESTER;

Grant succeeded.
```

```
SQL> ALTER USER TESTER QUOTA UNLIMITED ON USERS;

User altered.

SQL> ALTER USER TESTER QUOTA 100M ON USERS;

User altered.

SQL> SELECT tablespace_name, bytes, max_bytes FROM dba_ts_quotas WHERE username = 'C##TEST_USER';

TABLESPACE_NAME                    BYTES  MAX_BYTES
------------------------------ ---------- ----------
USERS                                   0  104857600
```

3. Connect as the new user.

```
SQL> CONNECT TESTER/test_password;

Connected.
```

4. Create a table

```
SQL> CREATE TABLE CUSTOMERS (ID NUMBER(5), NAME VARCHAR(42), CREDIT_LIMIT NUMBER(10));

Table created.
```

5. Add data to the table.

```
SQL> INSERT INTO CUSTOMERS VALUES (001, 'Rakesh Sharma', 10000);

1 row created.

SQL> INSERT INTO CUSTOMERS VALUES (002, 'Betty John', 20000);

1 row created.

SQL> INSERT INTO CUSTOMERS VALUES (003, 'T Ramchandran', 30000);

1 row created.

SQL> INSERT INTO CUSTOMERS VALUES (004, 'Amir Khan', 40000);

1 row created.

SQL> COMMIT;

Commit complete.
```

6. Encrypt a column.

```
SQL> ALTER TABLE CUSTOMERS MODIFY (CREDIT_LIMIT ENCRYPT);

Table altered.
```

7. List encrypted columns.

To do this you must connect as the sysdba user;

```
SQL> CONNECT sys/oracle as sysdba;
SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;

OWNER     TABLE_NAME  COLUMN_NAME   ENCRYPTION_ALG    SAL INTEGRITY_AL
--------- ----------- ------------- ----------------- --- -----------------------
TEST_USER CUSTOMERS   CREDIT_LIMIT  AES 192 bits key  YES SHA-1
```

2.8.1.1.1. Verification

1. Connect as the test_user

```
SQL> CONNECT TESTER/test_password;

Connected.
```

2. Get the encrypted data back.

```
SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;

CREDIT_LIMIT
------------
       10000
       20000
       30000
       40000
```

3. Close the wallet

   To do this you must connect as the sysdba user;

```
SQL> CONNECT sys/oracle as sysdba;

-- Multitenant
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "file:/opt/oracle/entrust/oracle.conf"
CONTAINER = ALL;

keystore altered.

-- Non-Multitenant
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "file:/opt/oracle/entrust/oracle.conf";

keystore altered.
```

4. Connect as the test_user

```
SQL> CONNECT TESTER/test_password;

Connected.
```

5. Now data retrieval should fail.

```
SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;
ERROR at line 1:
ORA-28365: wallet is not open
```

6. Works again after opening the key store.

   To do this you must connect as the sysdba user;

```
SQL> CONNECT sys/oracle as sysdba;

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

keystore altered.

-- Non-Multitenant
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "file:/opt/oracle/entrust/oracle.conf";

keystore altered.
```

7. Now the retrieval of the data should work.

   You must connect as the test_user first.

```
SQL> CONNECT TESTER/test_password;

Connected.

SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;

CREDIT_LIMIT
------------
       10000
       20000
       30000
       40000
```

2.8.1.2. Tablespace encryption

Data encryption keys are managed by Oracle. They are created as tablespaces are encrypted. The DEKs are wrapped with the master key and the wrapped keys are stored with the data. Oracle communicates with Key Management Vault to wrap / unwrap these data keys.

Make sure you are connected as sysdba.

1. Create encrypted tablespace, for example using AES256 algorithm

```
SQL> CREATE TABLESPACE SECURESPACE DATAFILE '/opt/oracle/oradata/CDB1/SECURE01.DBF' SIZE 150M ENCRYPTION
```

```
using 'AES256' DEFAULT STORAGE (ENCRYPT);

Tablespace created.
```

2. Create a table in encrypted tablespace.

```
SQL> CREATE TABLE EMPLOYEE (ID NUMBER(5),NAME VARCHAR(42),SALARY NUMBER(10)) TABLESPACE SECURESPACE;

Table created.
```

3. Insert data in the table.

```
SQL> INSERT INTO EMPLOYEE VALUES (001,'JOHN LENON',100000);

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES (002,'MARTINA HINGIS',200000);

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES (003,'R ATTENBOROUGH',350000);

1 row created.

SQL> COMMIT;

Commit complete.
```

4. Display the data.

```
SQL> SELECT * FROM EMPLOYEE;

        ID NAME                                          SALARY
---------- ---------------------------------------- ----------
         1 JOHN LENON                                   100000
         2 MARTINA HINGIS                               200000
         3 R ATTENBOROUGH                               350000
```

5. Display tablespaces.

```
SQL> SELECT t.name AS tablespace_name, e.encryptionalg AS encryption_algorithm FROM v$tablespace t JOIN v
$encrypted_tablespaces e ON t.ts# = e.ts#;

TABLESPACE_NAME                ENCRYPT
------------------------------ -------
SECURESPACE                    AES256
```

2.8.1.2.1. Verification

1. Get the encrypted data back

```
SQL> SELECT * FROM EMPLOYEE;
```

```
        ID NAME                                          SALARY
---------- ---------------------------------------- ----------
         1 JOHN LENON                                   100000
         2 MARTINA HINGIS                               200000
         3 R ATTENBOROUGH                               350000
```

2. Close the HSM wallet.

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

keystore altered.

-- Non-Multitenant
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "file:/opt/oracle/entrust/oracle.conf";

keystore altered.
```

3. Now data retrieval should fail.

```
SQL> SELECT * FROM EMPLOYEE;

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

4. Open the keystore.

   Data retrieval works again after opening the keystore.

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

keystore altered.

-- Non-Multitenant
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "file:/opt/oracle/entrust/oracle.conf";

keystore altered.
```

5. Retrieve the data - It should work!

```
SQL> SELECT * FROM EMPLOYEE;

        ID NAME                                          SALARY
---------- ---------------------------------------- ----------
         1 JOHN LENON                                   100000
         2 MARTINA HINGIS                               200000
         3 R ATTENBOROUGH                               350000
```

2.8.1.2.2. Online encryption of tablespace

If the tablespace is already created without enabling encryption, then the encryption can be enabled later, like this:

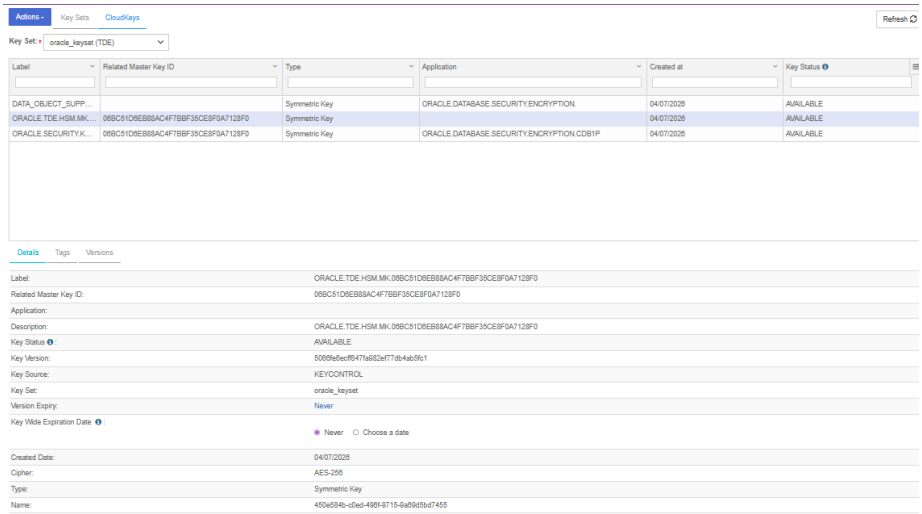
```
SQL> ALTER TABLESPACE plaintext_tablespace encryption online using 'AES192' encrypt;
```

2.8.2. Oracle TDE Keys in the Key Management Vault web GUI

The Oracle TDE keys are created by Oracle using the pkcs11 API directly. However, Key Management Vault shows these keys in the web GUI. There are a limited set of operations that can be performed from the web GUI.

In the Entrust Cryptographic Security Platform Key Management Vault Database Vault web GUI:

1. Navigate to **Cloudkeys**.
2. Select the **CloudKeys** tab.
3. For the **Key Set**, select the TDE keyset created by the TDE setup scripts.
4. The Keys that have been created by our encryption process in the previous section should be listed.



The keys starting with description:

- **ORACLE.TDE.HSM.MK** : are the TDE master keys created by Oracle. These are used to encrypt TDE data encryption keys.
- **ORACLE.SECURITY.KM.ENCRYPTION** : are generic keys managed by Oracle for TDE. They can be used to encrypt data or other keys.
- **DATA_OBJECT_SUPPORTED_IDEN** : these are objects used by Oracle for access control purposes

2.9. Migrate from software wallet to Key Management Vault

If the database is already encrypted with a software keystore, use the migration process.

Oracle requires both the software keystore and the HSM keystore to be open when transitioning from a software wallet to an HSM keystore. However, the database administrator cannot open both stores simultaneously. Instead, Oracle uses an auto-login wallet to open both keystores during migration. This applies to both primary and standby servers or clusters.

The following process guides the administrator to set up an auto-login wallet on the primary server and then copy the wallet files to the standby setup. These are mandatory steps, do not skip them.

1. On the standby server, stop the Managed Recovery Process (MRP) using `stop_redo_log_apply`.

   ```
   % ./encrypt.sh standby stop_redo_log_apply
   ```

   For an example of the full output for this command, see here.

2. On the primary server, run `migrate` and set up the auto-login wallet.

   ```
   % ./encrypt.sh migrate
   ```

   For an example of the full output for this command, see here.

3. Set up auto-login.

   ```
   % ./encrypt.sh setup_auto_login
   ```

   For an example of the full output for this command, see here.

4. Check the `WALLET_ROOT` and the location of the wallet files.

   ```
   % ./encrypt.sh status
   ```

   The `WALLET_ROOT` and wallet files appear in the output of the `encrypt.sh status` command.

   ```
   Current wallet_root     ------- /opt/oracle/oradata/CDB1/wallet

   .
   ```

```
.
.

Wallet files
------------------------
/opt/oracle/oradata/CDB1/wallet/tde/cwallet.sso
/opt/oracle/oradata/CDB1/wallet/tde/ewallet_2026041014532477.p12
/opt/oracle/oradata/CDB1/wallet/tde/ewallet_2026041017504798_tde_backup.p12
/opt/oracle/oradata/CDB1/wallet/tde/ewallet_2026041017505180_tde_backup.p12
/opt/oracle/oradata/CDB1/wallet/tde/ewallet_2026041017574123_tde_backup.p12
/opt/oracle/oradata/CDB1/wallet/tde/ewallet.p12
```

5. Copy `WALLET_ROOT` files from the primary wallet to the standby wallet.

   `WALLET_ROOT` is set to `/opt/oracle/oradata/CDB1/wallet`.

```
% scp $WALLET_ROOT/tde/* oracle@standby:$WALLET_ROOT/tde

cwallet.sso                                   100% 5160     2.8MB/s   00:00
ewallet_2026041014532477.p12                  100% 2555   965.1KB/s   00:00
ewallet_2026041017504798_tde_backup.p12       100% 3995     3.2MB/s   00:00
ewallet_2026041017505180_tde_backup.p12       100% 3995     2.9MB/s   00:00
ewallet_2026041017574123_tde_backup.p12       100% 3995     3.2MB/s   00:00
ewallet.p12
```

6. On the standby server, restart the database and then restart redo log apply.

   You want to make sure the MRP process is running. Connect to the database as sysdba and run the following commands:

```
% sqlplus / as sysdba

SQL> ALTER SYSTEM SET TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=HSM|FILE" scope=both;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE OPEN READ ONLY;
SQL> select DB_UNIQUE_NAME, SWITCHOVER_STATUS, DATABASE_ROLE, OPEN_MODE from V$DATABASE;

DB_UNIQUE_NAME                 SWITCHOVER_STATUS    DATABASE_ROLE    OPEN_MODE
------------------------------ -------------------- ---------------- --------------------
CDB1S                          NOT ALLOWED          PHYSICAL STANDBY READ ONLY

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT FROM SESSION;
```

7. Check the status.

```
% ./encrypt.sh status
```

   For an example of the full output for this command, see here.

8. Check that you can read data on the standby.

   Connect to the database as sysdba and run the following commands:

```
% sqlplus / as sysdba

SQL> SELECT * FROM EMPLOYEE;

        ID NAME                                          SALARY
---------- ---------------------------------------- ----------
         1 JOHN LENNON                                  100000
         2 MARTINA HINGIS                               200000
         3 R ATTENBOROUGH                               350000
```

2.10. Rotation of the TDE Key

As with the other key operations, rekeying is also initiated on the Oracle server. The Oracle server deactivates the existing key and creates a new master key. The encryption key is then wrapped using the new master key. Note that the older key is deactivated but not destroyed.

If the master key has to be rotated while there is an active Data Guard managed standby, the admin must first stop the redo log apply on the standby.

1. Stop the redo log apply on standby.

   ```
   % ./encrypt.sh standby stop_redo_log_apply
   ```

   For an example of the full output for this command, see here.

2. Rotate the key on Primary.

   You must execute this command on the primary node.

   ```
   > ./encrypt.sh rotate_key
   ```

   For an example of the full output for this command, see here.

3. Restart the redo log apply.

   Once the master key is rotated on primary, the admin should restart the redo log apply on the standby.

   ```
   % ./encrypt.sh standby restart_redo_log_apply
   ```

   For an example of the full output for this command, see here.

4. Check the status of the MRP process.

   When the key is rotated on the primary, the Managed Recovery Process (MRP) on the

standby does not get the key immediately. After restarting the redo log, check the status of MRP using the following command:

```
% ./encrypt.sh standby redo_log_status
```

The status appears in the following section of the output:

```
Managed Recovery Process
-------------------------
PROCESS   STATUS          THREAD#  SEQUENCE#
--------- ------------ ---------- ----------
MRP0      APPLYING_LOG          1         21
```

For an example of the full output for this command, see here.

If the MRP has not started applying log, then wait for a couple of seconds and then try to restart it again.

5. Check the status.

```
% ./encrypt.sh status
```

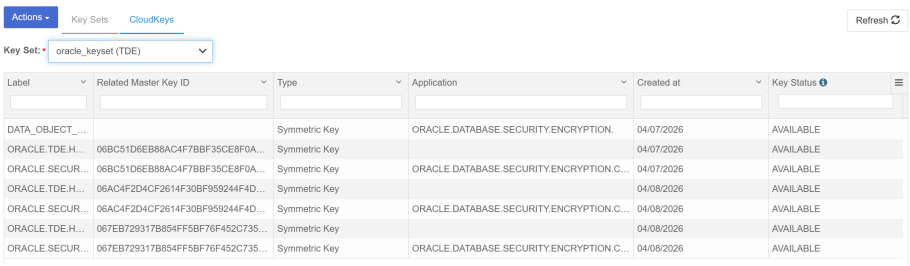
For an example of the full output for this command, see here.

2.10.1. Check the rotation of the keys in the Key Management Vault web GUI.

You can check rotation of the keys by using the Key Management Vault web GUI. You should see the new keys that have been created in the UI.

In the Cryptographic Security Platform Key Management Vault Database Vault web GUI:

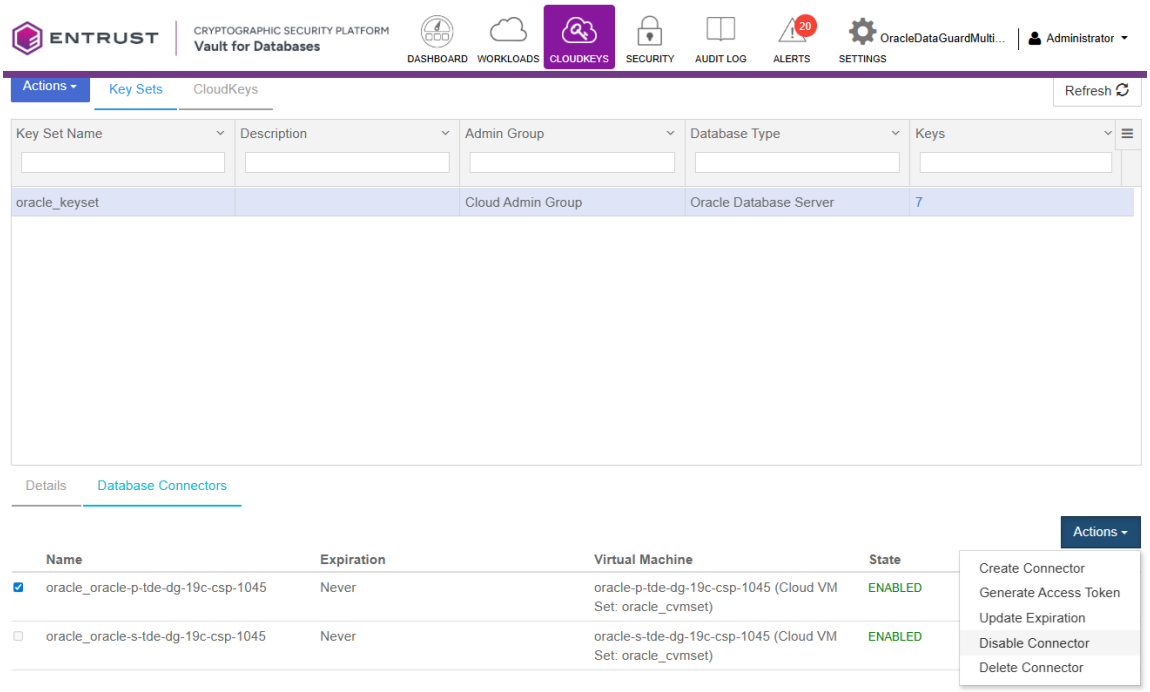
1. Navigate to **Cloudkeys**.
2. Select the **CloudKeys** tab.
3. For the **Key Set**, select the TDE keyset created by the TDE setup scripts.
4. You should see the new keys that have been created by the key rotation.



2.11. Disable or enable the Database Connector

2.11.1. Disable the Database Connector

1. Log in to the Key Management Vault 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**.



7. Select **Disable**.
8. Confirm that the state is DISABLED.
9. Return to the Oracle Server and login to SQL 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:

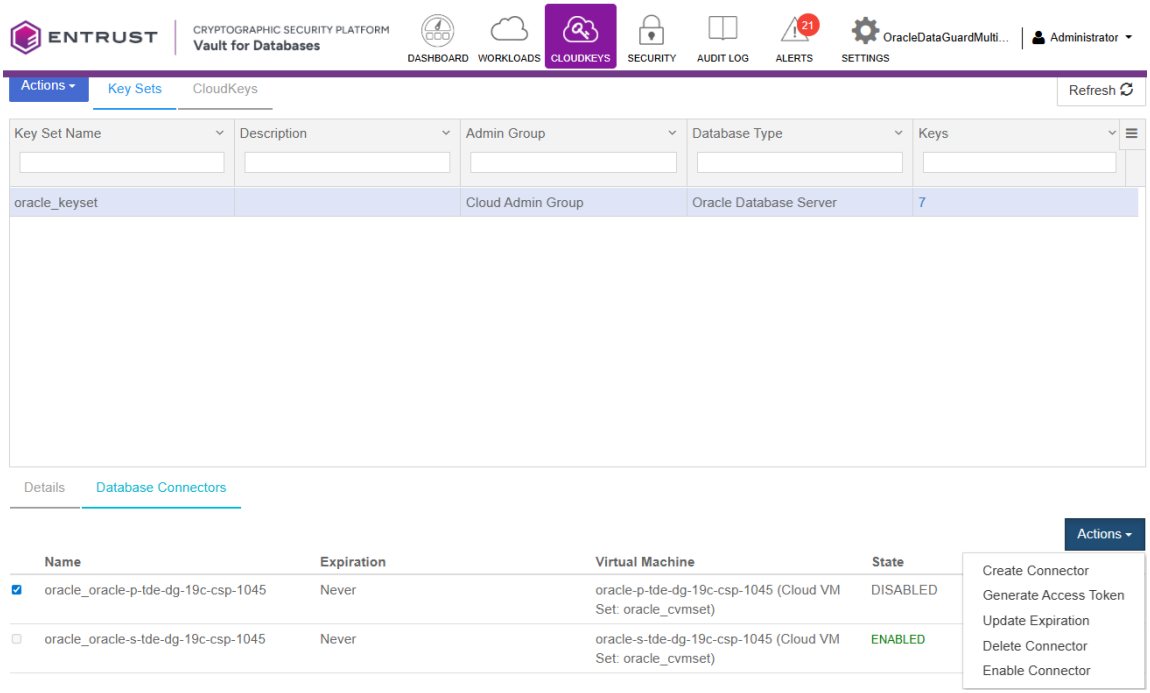
```
SQL> select CON_ID,WRL_TYPE,STATUS from V$ENCRYPTION_WALLET;

    CON_ID WRL_TYPE             STATUS
```

```
---------- -------------------- --------------------------------
         1 HSM                  CLOSED
         2 HSM                  CLOSED
         3 HSM                  CLOSED
         4 HSM                  CLOSED
```

2.11.2. Enable the Database Connector

1. Log in to the Key Management Vault 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**.



7. Open the keystore:

   Return to the Oracle Server and login as sysdba and run the following SQL command:

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

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
```

2.11.3. 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.

```
SQL> select * from employee;

        ID NAME                                          SALARY
---------- ---------------------------------------- ----------
         1 JOHN LENON                                   100000
         2 MARTINA HINGIS                               200000
         3 R ATTENBOROUGH                               350000
```

2.12. Reverse Migration from Key Management Vault to software wallet

In this section, we will migrate back from the CSP Vault keystore to the software wallet.

1. On Standby, stop Managed Recovery Process, i.e. stop redo log apply

   ```
   % ./encrypt.sh standby stop_redo_log_apply
   ```

   For an example of the full output for this command, see here.

2. On the primary, use reverse migrate. This will also set up auto-login.

   ```
   % ./encrypt.sh reverse_migrate
   ```

   For an example of the full output for this command, see here.

3. Add the config file as secret for a client HSM_PASSWORD.

   Add the secret to the software keystore. This secret represents the config file, with the client identified as HSM_PASSWORD. HSM_PASSWORD is an oracle defined client name that represents the HSM password as a secret in the software keystore.

   Connect to the database as sysdba and run the following commands:

   ```
   % sqlplus / as sysdba

   SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'file:/opt/oracle/entrust/oracle.conf' for client 'HSM_PASSWORD'
   identified by "mypassword" with backup;
   SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE IDENTIFIED BY "mypassword";
   ```

4. Copy WALLET_ROOT files to standby.

   Notice WALLET_ROOT is set to `/opt/oracle/oradata/CDB1/wallet`.

```
% scp $WALLET_ROOT/tde/* oracle@standby:$WALLET_ROOT/tde

cwallet.sso                                   100% 5160     2.8MB/s   00:00
ewallet_2026041014532477.p12                  100% 2555   965.1KB/s   00:00
ewallet_2026041017504798_tde_backup.p12       100% 3995     3.2MB/s   00:00
ewallet_2026041017505180_tde_backup.p12       100% 3995     2.9MB/s   00:00
ewallet_2026041017574123_tde_backup.p12       100% 3995     3.2MB/s   00:00
ewallet.p12
```

5. On Standby server, bounce the database and restart redo log apply.

   Connect to the database as sysdba and run the following commands:

```
% sqlplus / as sysdba

SQL> ALTER SYSTEM SET TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=HSM|FILE" scope=both;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE OPEN READ ONLY;
SQL> select DB_UNIQUE_NAME, SWITCHOVER_STATUS, DATABASE_ROLE, OPEN_MODE from V$DATABASE;

DB_UNIQUE_NAME                 SWITCHOVER_STATUS    DATABASE_ROLE    OPEN_MODE
------------------------------ -------------------- ---------------- --------------------
CDB1S                          NOT ALLOWED          PHYSICAL STANDBY READ ONLY

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT FROM SESSION;
```

6. Check the status:

```
% ./encrypt.sh status
```

   For an example of the full output for this command, see here.

2.13. Backup or restore

When using Key Management Vault, when doing a backup and restore, make sure Entrust policy agent is setup on the restore node as well. Before recovering the restored database, the HSM keystore has to be opened. Refer the Oracle documentation for details of RMAN based backup /restore.

https://docs.oracle.com/en/database/oracle/oracle-database/19/bradv/part-overview-backup-recovery.html

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/oracle.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. Example command output

4.1. encrypt.sh

4.1.1. encrypt

```
% ./encrypt.sh encrypt

Logging debug trace and output to: ./trace/encrypt_5.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf

Database CDB1 is already running

Oracle UNQNAME ---------- CDB1P

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PRIMARY

Encrypting database CDB1P with TDE using KeyControl
CDB1P is open read write

Set WALLET_ROOT now

WALLET ROOT (/opt/oracle/oradata/CDB1/wallet) ?

Creating WALLET_ROOT "/opt/oracle/oradata/CDB1/wallet" -------
Done

Set WALLET_ROOT to "/opt/oracle/oradata/CDB1/wallet" -------
Done

Restarting database CDB1P -------
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers               7639040 bytes
Database mounted.
Database opened.
Done

Set TDE_CONFIGURATION to HSM -------
Done

Restarting database CDB1P -------
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
```

```
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers               7639040 bytes
Database mounted.
Database opened.
Done

Opening HSM (KeyControl) KeyStore -------
Done

encrypt with KeyControl -------
Done

Restarting database CDB1P -------
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers               7639040 bytes
Database mounted.
Database opened.
Done

Opening HSM (KeyControl) KeyStore -------
Done
Database report temporarily generated in file: /opt/hcs/tde_reports/CDB1_04092026_115029.report
It will be pushed to KeyControl and removed from local system.

Encryption of database CDB1P with TDE using KeyControl completed
```

4.1.2. migrate

```
% ./encrypt.sh migrate

Logging debug trace and output to: ./trace/encrypt_5.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf

Database CDB1 is already running

Oracle UNQNAME ---------- CDB1P

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PRIMARY

Migrating database CDB1P from Software Wallet to KeyControl
CDB1P is open read write

Removing Auto login Wallet -------
No auto login wallet found
Done

Restarting database CDB1P -------
```

```
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers                7639040 bytes
Database mounted.
Done

Set TDE_CONFIGURATION to FILE -------
Done

Opening Software Wallet and database in Read Write mode -------
Done

Backup Software Wallet with tag tde_backup -------
Done

Set TDE_CONFIGURATION to HSM|FILE -------
Done

Migrate to KeyControl -------
Done

Restarting database CDB1P -------
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers                7639040 bytes
Database mounted.
Done

Opening HSM (KeyControl) KeyStore and database in Read Write mode -------
Done
Database report temporarily generated in file: /opt/hcs/tde_reports/CDB1_04102026_134932.report
It will be pushed to KeyControl and removed from local system.

Migration of database CDB1P from Software Wallet to KeyControl completed
```

4.1.3. reverse_migrate

```
% ./encrypt.sh reverse_migrate

Logging debug trace and output to: ./trace/encrypt_8.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf

Database CDB1 is already running

Oracle UNQNAME ---------- CDB1P
```

```
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PRIMARY

Reversing migration from KeyControl to Software Wallet for database CDB1P
CDB1P is open read write

Removing Auto login Wallet -------
Creating backup /opt/oracle/oradata/CDB1/wallet/tde/cwallet_04102026_144854.sso and removing auto login wallet
/opt/oracle/oradata/CDB1/wallet/tde/cwallet.sso
Done

Restarting database CDB1P -------
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers                7639040 bytes
Database mounted.
Done

Set TDE_CONFIGURATION to FILE -------
Done

Opening Software Wallet and database in Read Write mode -------
Done

Remove Secret from Software Wallet -------
Done

Restarting database CDB1P -------
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers                7639040 bytes
Database mounted.
Done

Set TDE_CONFIGURATION to HSM -------
Done

Opening HSM (KeyControl) KeyStore and database in Read Write mode -------
Done

Set TDE_CONFIGURATION to FILE|HSM -------
Done

Reverse Migrate to Software Wallet -------
Done

Restarting database CDB1P -------
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
```

```
Database Buffers         1879048192 bytes
Redo Buffers               7639040 bytes
Database mounted.
Done

Set TDE_CONFIGURATION to FILE -------
Done

Opening Software Wallet and database in Read Write mode -------
Done
Database report temporarily generated in file: /opt/hcs/tde_reports/CDB1_04102026_144854.report
It will be pushed to KeyControl and removed from local system.

Reversing migration from KeyControl to Software Wallet completed
```

4.1.4. rotate_key

```
> ./encrypt.sh rotate_key

Logging debug trace and output to: ./trace/encrypt_7.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf

Database CDB1 is already running

Oracle UNQNAME ---------- CDB1P

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PRIMARY

Rotating TDE master key for database CDB1P using KeyControl
Database report temporarily generated in file: /opt/hcs/tde_reports/CDB1_04092026_134706_prerotate.report
It will be pushed to KeyControl and removed from local system.

Database report temporarily generated in file: /opt/hcs/tde_reports/CDB1_04092026_134706.report
It will be pushed to KeyControl and removed from local system.

Rotation of TDE master key for database CDB1P completed
```

4.1.5. setenv

```
% ./encrypt.sh setenv

Logging debug trace and output to: ./trace/encrypt_1.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf

Database SID (CDB1) ?

ORACLE_BASE (/opt/oracle) ?
```

```
ORACLE_HOME (/opt/oracle/product/19c/dbhome_1) ?

Software Wallet Password () ?

Type Password again (**********) ?

Access token file (/opt/oracle/entrust/oracle.conf) ?

Successfully set environment variables for TDE scripts in ./oracle.env
Updated env cache ./oracle.tab
```

4.1.6. setup_auto_login

```
% ./encrypt.sh setup_auto_login

Logging debug trace and output to: ./trace/encrypt_6.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf

Database CDB1 is already running

Oracle UNQNAME ---------- CDB1P

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PRIMARY

Configuring auto login for KeyControl for database CDB1P
CDB1P is open read write

Set TDE_CONFIGURATION to HSM -------
Done

Closing HSM (KeyControl) KeyStore -------
Done

Set TDE_CONFIGURATION to FILE -------
Done

Opening Software Wallet -------
Done

Add Secret to Software Wallet -------
Done

Restarting database CDB1P -------
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers                7639040 bytes
Database mounted.
Done
```

```
Create Auto login Keystore -------
Done

Set TDE_CONFIGURATION to HSM|FILE -------
Done

Restarting database CDB1P -------
ORA-01109: database not open

Database dismounted.
ORACLE instance shut down.
ORACLE instance started.
Total System Global Area 2432695832 bytes
Fixed Size                  9137688 bytes
Variable Size             536870912 bytes
Database Buffers         1879048192 bytes
Redo Buffers                7639040 bytes
Database mounted.
Database opened.
Done

Opening All PDBs -------
Done
Database report temporarily generated in file: /opt/hcs/tde_reports/CDB1_04102026_135719.report

It will be pushed to KeyControl and removed from local system.

Configuration of auto login for KeyControl for database CDB1P completed
```

4.1.7. standby redo_log_status

```
% ./encrypt.sh standby redo_log_status

Logging debug trace and output to: ./trace/encrypt_7.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf

Database CDB1 is already running

Oracle UNQNAME ---------- CDB1S

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PHYSICAL STANDBY

Checking status of managed recovery process on standby server for database CDB1S

Managed Recovery Process
-------------------------
PROCESS   STATUS          THREAD#  SEQUENCE#
--------- ------------ ---------- ----------
MRP0      APPLYING_LOG          1         21
```

4.1.8. standby restart_redo_log_apply

```
% ./encrypt.sh standby restart_redo_log_apply

Logging debug trace and output to: ./trace/encrypt_4.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf


Database CDB1 is already running

Oracle UNQNAME ---------- CDB1S

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PHYSICAL STANDBY

Restarting managed recovery process on standby server for database CDB1S
Done
```

4.1.9. standby setup

```
% ./encrypt.sh standby setup

Logging debug trace and output to: ./trace/encrypt_1.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf

Database CDB1 is already running

Oracle UNQNAME ---------- CDB1S

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PHYSICAL STANDBY

.
.
.

Opening HSM (KeyControl) KeyStore -------
Done

recover database -------
Done

Opening database read only -------
Done

Setting DataGuard mapping for TDE KeyStore -------
Successfully updated config file /opt/oracle/entrust/oracle.conf
Done

Alter database to MOUNTED mode -------
Done
```

```
Restarting Redo Log apply -------
Done
Database report temporarily generated in file: /opt/hcs/tde_reports/CDB1_04092026_112443.report
It will be pushed to KeyControl and removed from local system.

Setup of TDE parameters on standby server for database CDB1S completed
```

4.1.10. standby stop_redo_log_apply

```
% ./encrypt.sh standby stop_redo_log_apply

Logging debug trace and output to: ./trace/encrypt_12.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf


Database CDB1 is already running

Oracle UNQNAME ---------- CDB1S

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PHYSICAL STANDBY

Stopping managed recovery process on standby server for database CDB1S
Done
```

4.1.11. status

```
% ./encrypt.sh status

Logging debug trace and output to: ./trace/encrypt_2.trc

Oracle SID -------------- CDB1

Using configuration file: ./entrust.conf
Using environment file:   ./oracle.env
Using access token file:  /opt/oracle/entrust/oracle.conf


Database CDB1 is already running

Oracle UNQNAME ---------- CDB1P

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Database Role: PRIMARY

db_unique_name          ------- CDB1P
Multitenant database    ------- YES
Database type           ------- Single instance

tde_configuration       -------
db_create_file_dest     -------
Current wallet_root     -------
```

```
Calculated wallet_root  ------- /opt/oracle/oradata/CDB1/wallet

Database encryption wallet  ------- need to open wallet/s to check
Auto Login enabled          ------- NO

Database state
------------------------
Database Name   Open Mode                Database Role             Switchover Status
--------------- ------------------------ ------------------------- ------------------------
CDB1            READ WRITE               PRIMARY                   TO STANDBY

SHOW PDBs
------------------------
PDB Name        CON_ID OPEN_MODE  RES
--------------- ------ ---------- ---
PDB$SEED             2 READ ONLY  NO
CDB1PDB1             3 MOUNTED
CDB1PDB2             4 MOUNTED

Encryption Wallets
------------------------
PDB Name   Status               WRL_TYPE   WALLET_OR Wallet Type  KEYSTORE WRL_PARAMETER
---------- -------------------- ---------- --------- ------------ --------
---------------------------------------
CDB$ROOT   NOT_AVAILABLE        FILE       SINGLE    UNKNOWN      NONE     /opt/oracle/admin/CDB1P/wallet
CDB1PDB1   NOT_AVAILABLE        FILE       SINGLE    UNKNOWN      UNITED
CDB1PDB2   NOT_AVAILABLE        FILE       SINGLE    UNKNOWN      UNITED
PDB$SEED   NOT_AVAILABLE        FILE       SINGLE    UNKNOWN      UNITED

Current Database master encryption key/s in use
-----------------------------------------------

Encrypted Tablespaces
---------------------------------

TDE Master keys in open wallets
---------------------------------

Services
------------------------
cdb1pdb1
SYS$BACKGROUND
SYS$USERS
CDB1P.ncipher.com
CDB1XDB
cdb1pdb2

Datafiles
------------------------
/opt/oracle/oradata/CDB1/system01.dbf
/opt/oracle/oradata/CDB1/sysaux01.dbf
/opt/oracle/oradata/CDB1/undotbs01.dbf
/opt/oracle/oradata/CDB1/pdbseed/system01.dbf
/opt/oracle/oradata/CDB1/pdbseed/sysaux01.dbf
/opt/oracle/oradata/CDB1/users01.dbf
/opt/oracle/oradata/CDB1/pdbseed/undotbs01.dbf
/opt/oracle/oradata/CDB1/CDB1PDB1/system01.dbf
/opt/oracle/oradata/CDB1/CDB1PDB1/sysaux01.dbf
/opt/oracle/oradata/CDB1/CDB1PDB1/undotbs01.dbf
/opt/oracle/oradata/CDB1/CDB1PDB1/users01.dbf
/opt/oracle/oradata/CDB1/CDB1PDB2/system01.dbf
/opt/oracle/oradata/CDB1/CDB1PDB2/sysaux01.dbf
/opt/oracle/oradata/CDB1/CDB1PDB2/undotbs01.dbf
/opt/oracle/oradata/CDB1/CDB1PDB2/users01.dbf

Wallet files
------------------------
```

```
WALLET_ROOT not set. No wallet files found

ASM disk group
------------------------

ASM check
------------------------
CDB1P is not using ASM
```

4.2. setup.sh

4.2.1. first entrust.conf

```
% sudo ./setup.sh first entrust.conf

Logging debug trace and output to: ./trace/setup.trc
First node configuration in Oracle RAC

Enter password for xxxxx.xxxxxx@entrust.com:

Downloading hicli API from xxx.xxx.xxx.xxx
.
.
.

Create Access Token
Successfully created access token

Saving access token in file /opt/oracle/entrust/oracle.conf
Set permission of access token so that user "oracle" can access it
Successfully set permissions

Create link to Entrust pkcs11 library in /opt/oracle/extapi/64/hsm/entrust

Successfully linked pkcs11 library

Setup complete
```

4.2.2. other entrust.conf

```
% sudo ./setup.sh other entrust.conf

Logging debug trace and output to: ./trace/setup.trc
Other node configuration

Enter password for xxxxx.xxxxx@entrust.com:

Downloading hicli API from xxx.xxx.xxx.xxx

Successfully downloaded and extracted Entrust KeyControl API

.
.
.

Create Access Token
Successfully created access token
```

```
Saving access token in file /opt/oracle/entrust/oracle.conf
Set permission of access token so that user "oracle" can access it
Successfully set permissions

Create link to Entrust pkcs11 library in /opt/oracle/extapi/64/hsm/entrust
Successfully linked pkcs11 library

Setup complete
```

Chapter 5. Additional resources and related products

5.1. Entrust CSP Key Manager

5.2. Entrust products

5.3. nShield product documentation