THALES

Oracle Database

INTEGRATION GUIDE

THALES LUNA HSM

THALES DATA PROTECTION ON DEMAND

Document Information

Document Part Number	007-008670-001
Release Date	14 October 2020

Revision History

Revision	Date	Reason
BK	14 October 2020	Update

CONTENTS

CHAPTER 1: Getting Started

This chapter covers the following topics:

About Luna HSMs and Oracle TDE

Thales Luna HSMs come as on premise hardware HSMs, widely known as Luna HSMs and a cloud offering, HSM on Demand Service.

Oracle TDE provides the infrastructure necessary for implementing encryption. It allows you to encrypt sensitive data stored in application table columns (such as credit card numbers.) or application tablespaces. Oracle TDE uses the Luna HSMs or HSM on Demand service to secure the master encryption key for the following reasons:

> The HSMs store the master encryption keys used for transparent data encryption, and the master encryption key is never exposed in insecure memory.
> The HSMs provide more secure computational storage.
> The HSMs are a more secure alternative to the Oracle wallet.

Supported HSM Devices and Services

Below is the list of the supported HSMs:

Thales Luna HSM: Thales Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Thales Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing.

The Thales Luna HSM on premise offerings include the Thales Luna Network HSM, Thales PCIe HSM, and Thales Luna USB HSMs.

Thales DPOD: Thales Data Protection on Demand (DPoD) is a cloud-based platform that provides on-demand HSM and Key Management services through a simple graphical user interface. With DPOD,

security is simple, cost effective, and easy to manage because there is no hardware to buy, deploy, and maintain. As an Application Owner, you click and deploy services, generate usage reports and maintain the required services.

Certified Platforms

The integration of Oracle Database with Luna HSM is certified on the following operating systems:

Oracle Database	Platforms
Oracle Database 19C	Red Hat Enterprise Linux Windows Server 2019 Windows Server 2016 AIX Solaris x86
Oracle Database 18C	Red Hat Enterprise Linux Windows Server 2019 Windows Server 2016 AIX Solaris Sparc Solaris x86
Oracle Database 12C	Windows Server 2016 Windows Server 2012 R2 Red Hat Enterprise Linux Solaris Sparc Solaris x86 AIX Oracle Linux

NOTE: If you are using Oracle Database 11g R1/R2, refer to the previous version of the Oracle Database Integration Guide (OracleDatabase_LunaHSM_IntegrationGuide_RevBC).

NOTE: DPoD supports only Windows and Linux platforms for Oracle Database integration.

Multiple Database Support for Single Partition

Oracle Version	Partition Configuration	Key Lifecycle	Encryption wallet creation
12.1.0.1	Requires a unique dedicated partition per Oracle database/cluster	New Install	TDE and TSE keys are created
		Key Migrate	New TDE key is created but the TSE key remains the same
12.1.0.2 12.2.0.1 18.3.0.0 19.3.0.0	Supports a unique dedicated partition per Oracle database/cluster and also a single shared partition configuration for all Oracle databases/clusters	New Install	Only TDE key is created
		Key Migrate	New TDE key is created

Configuring Thales Luna HSM

If you are using a Thales Luna HSM, complete the following steps:

1. Verify the HSM is set up, initialized, provisioned, and ready for deployment. Refer to the *Thales Luna HSM Product Documentation* for more information.
2. Create a partition on the HSM that will be later used by Oracle Database.
3. Register a client for the system and assign the client to the partition for creating an NTLS connection. Initialize the Crypto Officer role for the registered partition.
4. Ensure that each partition is successfully registered and configured. The command to see the registered partitions is:

```
# /usr/safenet/lunaclient/bin/lunacm

LunaCM v7.2.0-220. Copyright (c) 2006-2017 SafeNet.

Available HSMs:

Slot Id ->              0

Label ->                Oracle

Serial Number ->        1238712343066

Model ->                LunaSA 7.2.0

Firmware Version ->     7.2.0

Configuration ->        Luna User Partition With SO (PED) Key Export With
Cloning Mode

Slot Description ->     Net Token Slot
```

NOTE: Follow the *Thales Luna HSM Product Documentation* for detailed steps for creating the NTLS connection, initializing the partitions, and initializing the Security Officer, Crypto Officer, and Crypto User roles.

Controlling User Access to the HSM

> **NOTE:** This section is applicable only for Linux users.

By default, only the root user has access to the HSM. You can specify a set of non-root users that are permitted to access the HSM, by adding them to the **hsmusers** group. The client software installation automatically creates the **hsmusers** group. The **hsmusers** group is retained when you uninstall the client software, allowing you to upgrade the software while retaining your **hsmusers** group configuration.

Adding a user to hsmusers group

To allow non-root users or applications access to the HSM, assign the user to the **hsmusers** group. The users you assign to the **hsmusers** group must exist on the client workstation. To add a user to hsmusers group:

1. Ensure that you have **sudo** privileges on the client workstation.
2. Add a user to the hsmusers group.

   ```
   sudo gpasswd --add <username> hsmusers
   ```

   Where `<username>` is the name of the user you want to add to the hsmusers group.

Removing a user from hsmusers group

To remove a user from hsmusers group:

1. Ensure that you have sudo privileges on the client workstation.
2. Remove a user from the hsmusers group.

   ```
   sudo gpasswd -d <username> hsmusers
   ```

   Where `<username>` is the name of the user you want to remove from the **hsmusers** group. You must log in again to see the change.

   > **NOTE:** The user you delete will continue to have access to the HSM until you reboot the client workstation.

Provisioning HSM on Demand service

This service enables your client machine to access an HSM application partition for storing cryptographic objects used by your applications. Application partitions can be assigned to a single client, or multiple clients can be assigned to, and share, a single application partition. You need to provision your application partition by initializing the following roles:

- **Security Officer (SO)** - Responsible for setting the partition policies and for creating the Crypto Officer.
- **Crypto Officer (CO)** - Responsible for creating, modifying, and deleting crypto objects within the partition. The CO can use the crypto objects and create an optional, limited-capability role called Crypto User that can use the crypto objects but cannot modify them.
- **Crypto User (CU)** – An optional role that can use crypto objects while performing cryptographic operations.

NOTE: Refer the “Thales Data Protection on Demand Application Owner Quick Start Guide” for configuring the HSM on Demand service and creating a service client.

The HSM service client package is a zip file containing system information needed to connect your client machine to an existing HSM on Demand service.

To Configure DPoD HSM on Demand service with/without Luna Client

HSM on Demand Service can be configured in the following scenarios:

- **User wants to use DPoD Client to access service partition:** Execute steps 1, 2, 3, 4 and 10 below.
- **User wants to use Luna Client to access the service partition:** Execute steps 1-10 below.
- **User wants to use existing Luna Client to access the service partition in Hybrid mode with Luna Partition:** Execute steps 1-10 below.

NOTE: Last two scenarios are supported for Universal Client only, from Luna Client v10.1.0 onwards.

To configure DPoD HSM on Demand service:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the DPoD client install directory.

   [Windows]

   ```
   cvclient-min.zip
   ```

   [Linux]

   ```
   cvclient-min.tar
   ```

   ```
   # tar -xvf cvclient-min.tar
   ```
4. Run the provided script to create a new configuration file containing information required by the HSMoD service.

   [Windows]

   Right-click **setenv.cmd** and select **Run as Administrator**.

   [Linux]

   Source the **setenv** script.

   ```
   # source ./setenv
   ```

NOTE: Run the LunaCM utility available in the DPoD client and verify the service partition is listed. If you need to configure DPoD service partition with existing Luna Client follow further steps.

5. Copy the server and partition certificates from the DPoD client directory to your Luna client certificates directory:

   DPoD Certificates:

   ```
   server-certificate.pem
   partition-ca-certificate.pem
   partition-certificate.pem
   ```

   LunaClient Certificate Directory:

   [Windows default]

   ```
   C:\Program Files\Safenet\Lunaclient\cert\
   ```

   [Linux default]

   ```
   /usr/safenet/lunaclient/cert/
   ```

6. Open the configuration file from the DPoD client directory and copy the **XTC** and **REST** section.

   [Windows]

   ```
   crystoki.ini
   ```

   [Linux]

   ```
   Chrystoki.conf
   ```

7. Edit the Luna Client configuration file and add the **XTC** and **REST** section. In both sections you need to change only server and partition certificates path from step 5. Do not change any other entries provided in **XTC** and **REST** section.

   ```
   [XTC]
   . . .
   PartitionCAPath=<LunaClient_cert_directory>\partition-ca-certificate.pem
   PartitionCertPath00=<LunaClient_cert_directory>\partition-certificate.pem
   . . .

   [REST]
   . . .
   SSLClientSideVerifyFile=<LunaClient_cert_directory>\server-certificate.pem
   . . .
   ```

8. Edit the following entry from the **Misc** section and update the correct path for the **plugins** directory:

   ```
   Misc]
   PluginModuleDir=<LunaClient_plugins_directory>
   ```

   [Windows Default]

   ```
   C:\Program Files\Safenet\Lunaclient\plugins\
   ```

   [Linux Default]

   ```
   /usr/safenet/lunaclient/plugins/
   ```

   Save the configuration file. If you wish, you can now safely delete the extracted DPoD client directory.

9. Reset the **ChrystokiConfigurationPath** environment variable and point back to the location of the Luna Client configuration file.

   [Windows]

   In the Control Panel, search for "environment" and select **Edit the system environment variables**. Click **Environment Variables**. In both list boxes for the current user and system variables, edit **ChrystokiConfigurationPath** and point to the **crystoki.ini** file in the Luna client install directory.

   [Linux]

   Either open a new shell session, or export the environment variable for the current session pointing to the location of the **Chrystoki.conf** file:

   ```
   # export ChrystokiConfigurationPath=/etc/
   ```

10. Run the **LunaCM** utility and verify the service partition is listed. If you already have a Luna Partition before configuring the DPoD service partition, both Luna and DPoD service partition will be listed.

Constraints on HSM on Demand Services

HSM on Demand Service in FIPS mode

HSMoD services operate in a FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, ensure you enable the **Allow non-FIPS approved algorithms** check box when configuring your HSM on Demand service. The FIPS mode is enabled by default. Refer to the *Mechanism List* in the *SDK Reference Guide* for more information about available FIPS and non-FIPS algorithms.

Verify HSM on Demand <slot value>

LunaCM commands work on the current slot. If there is only one slot, then it is always the current slot. If you are completing an integration using HSMoD services, you need to verify which slot on the HSMoD service you send commands to. If there is more than one slot, then use the slot set command to direct a command to a specified slot. You can use slot list to determine which slot numbers are in use by which HSMoD service.

Using Thales HSM in FIPS Mode

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for a FIPS-compliant HSM. If you are using the Luna HSM or HSM on Demand service in FIPS mode, you have to make the following change to the configuration file:

```
[Misc]

RSAKeyGenMechRemap=1
```

The above setting redirects the older calling mechanism to a new approved mechanism when Luna HSM or HSMoD is in FIPS mode.

> **NOTE:** For Universal Client, above setting is not required. This setting is applicable for Luna Client 7.x only.

NOTE: This remapping is automatic if you are using Luna HSM Client 10.1 and above, and the configuration file entry is ignored.

Setting Flags for Luna HSM on Window Server

If you are using a Luna HSM 7.x with Windows Server, open the **Crystoki.ini** file in a text editor and set the following two flags in the Misc section:

```
[Misc]
AllowMultipleInitialize=1
AllowMultipleFinalize=1
```

Configuring Luna HSM HA (High-Availability)

Please refer to the Luna HSM documentation for HA steps and details regarding configuring and setting up two or more HSM appliances on Windows and UNIX systems. You must enable the HAOnly setting in HA for failover to work so that if primary stop functioning for some reason, all calls automatically routed to secondary till primary starts functioning again.

CHAPTER 2: Integrating Luna HSM with Oracle Database

This chapter covers the following topics:

> Setting up Luna HSM Client for Transparent Data Encryption

> Configuring a Master Encryption Key for HSM-based encryption

> Working with Pluggable Databases (PDB)

> Storing Oracle Database Secrets in a hardware keystore

Setting up Luna HSM Client for Transparent Data Encryption

Copying the Luna HSM PKCS#11 Library

Copy the Luna HSM PKCS#11 library to the following directory structure to allow the Oracle database to access the cryptographic library.

Linux/Solaris/AIX	`"/opt/oracle/extapi/[32,64]/hsm/{Vendor}/{Version}/libXX.ext"`
Windows	`"%SYSTEMDRIVE%\oracle\extapi\[32,64]\hsm\{Vendor}\{Version}\libXX.ext"`

For example: 64-bit RHEL (/opt/oracle/extapi/64/hsm/safenet/7.x.x/libCryptoki2_64.so)

Parameter	Definition
[32,64]	Specifies whether the supplied binary is 32-bits or 64-bits.
[vendor]	Specifies the name of the vendor supplying the library.
[version]	Refers to the version number of the library. **NOTE:** The version number should be in the format [number.number.number]. The API name requires no special format. The XX must be prefixed with the word lib.
.ext	The extension must be replaced by the extension of the library file.

NOTE: Only one PKCS#11 library is supported at a time.

Granting Read/Write Permissions to the Oracle User

Grant the required read/write permissions to the Oracle user. Export the following variables:

```
export ORACLE_SID=orcl
export ORACLE_BASE=/u01/app/oracle (oracle installation directory)
export ORACLE_HOME=$ORACLE_BASE/product/<OracleDB_Version>/dbhome_1
export PATH=$PATH:$ORACLE_HOME/bin
export TNS_ADMIN=$ORACLE_HOME/network/admin
```

Configuring a Master Encryption Key for HSM-based encryption

To start using HSM-based encryption, you require a master encryption key that will be stored inside the HSM. The master encryption key is used to encrypt or decrypt the Oracle database table columns or tablespace using encryption keys stored inside the HSM. Using the HSM and protecting the Master Encryption Key involves the following scenarios:

> Generating the Master Encryption Key directly on the HSM

> Migrating Master Encryption Key from Software Keystore to Hardware Keystore

> Configuring Auto-login for Hardware Keystore

Generating the Master Encryption Key directly on the HSM

NOTE: It is assumed that no software or HSM based wallet is yet created.

To set up Oracle to create the master encryption key directly on the HSM

1. Refer to Appendix B and complete the hardware keystore configuration based on the installed Oracle Database version.

> Configure Hardware Keystore for Oracle Database 12c

> Configure Hardware Keystore for Oracle Database 18c and 19c

2. Start the database:

   ```
   $ sqlplus / as sysdba
   ```

   If the database is not yet started, you can start it using:

   ```
   SQL> startup;
   ```

3. Grant the **ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege to **SYSTEM** and any user that you want to use.

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

4. Connect to the database as 'system'.

   ```
   SQL> connect system/<password>
   ```

NOTE: Password for 'system' is set during Oracle installation. All dbapasswords throughout this document have been set to "temp123#".

5. Run the **ADMINISTER KEY MANAGEMENT SQL** statement to open the hardware keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<hsm_partition_password>;
```

6. Set the master encryption key in the hardware keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY
<hsm_partition_password>;
```

You can see the HSM partition contents to verify the generated keys on an HSM by executing **partition contents** in lunacm.

```
-------------------------------------------------------------------------------
-----------------------------------------------

Partition Name: part7

Partition SN: 152042028

Storage (Bytes): Total=102701, Used=1848, Free=100853

Number objects: 5

Object Label: ORACLE.TDE.HSM.MK.0661286A8C71864F2ABF7891D044154D9A

Object Type: Symmetric Key

Object Label: DATA_OBJECT_SUPPORTED_IDEN

Object Type: Data

Object Label:

ORACLE.SECURITY.KM.ENCRYPTION.303636313238364138433731383634463241424637383
9314430343431353

4443941

Object Type: Data

Object Label: DATA_OBJECT_SUPPORTED_IDEN

Object Type: Data

Object Label: ORACLE.TSE.HSM.MK.072AC159D9153C4FF0BF3BF931ED9693850203

Object Type: Symmetric Key
```

NOTE: In case a network latency is observed and heartbeat check fails with DPoD, Refer to Troubleshooting Tips Problem 4

To verify the master encryption key is encrypting the Oracle database

1. Create a CUSTOMERS table in the database.

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

2. Enter some values in the CUSTOMERS table.

```
SQL> INSERT INTO CUSTOMERS VALUES (001, 'George Bailey', 10000);
SQL> INSERT INTO CUSTOMERS VALUES (002, 'Denial Vettory', 20000);
SQL> INSERT INTO CUSTOMERS VALUES (003, 'MS Dhoni', 30000);
SQL> INSERT INTO CUSTOMERS VALUES (004, 'Shahid Afridi', 40000);
```

3. Encrypt the 'CREDIT_LIMIT' column of the 'CUSTOMERS' table.

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

4. With the next command, the values listed in the encrypted column are returned in clear text; Transparent Data Encryption decrypts them automatically.

```
SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;
```

5. The next command lists encrypted columns in your database.

```
SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;
```

6. Finally, this view contains information about the hardware keystore itself:

```
SQL> SELECT * FROM V$ENCRYPTION_WALLET;
```

7. Create an encrypted tablespace.

```
SQL> CREATE TABLESPACE SECURESPACE DATAFILE
'/u01/app/oracle/oradata/orcl/SECURE01.DBF' SIZE 150M ENCRYPTION DEFAULT
STORAGE (ENCRYPT);
```

8. Create a table in the tablespace.

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

9. Insert some values in EMPLOYEE table.

```
SQL> INSERT INTO EMPLOYEE VALUES (001,'JOHN SMITH',15000);
SQL> INSERT INTO EMPLOYEE VALUES (002,'SCOTT TIGER',25000);
SQL> INSERT INTO EMPLOYEE VALUES (003,'DIANA HAYDEN',35000);
```

10. Display the contents of the EMPLOYEE table with the following command:

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

11. Close the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<hsm_partition_password>;
```

12. After closing the keystore execute the command to display the contents again:

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

If the keystore is closed, you will get the following error that means you cannot list the contents of EMPLOYEE table:

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

13. Open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<hsm_partition_password>;

SQL> exit
```

Creating a TDE Master Encryption Key for Later Use and Activating a TDE Master Encryption Key

The **CREATE KEY** clause of the **ADMINISTER KEY MANAGEMENT** statement can create a TDE master encryption key to be activated at a later date. This method of TDE master encryption key creation is useful in a multitenant environment when you must re-create the TDE master encryption keys.

To create a TDE Master Encryption Key for later use

1. Log in to the database instance as a user who has been granted the **ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege.

   ```
   SQL> connect system/<password>
   ```

2. Open the keystore.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
   <hsm_partition_password>;
   ```

3. Create the TDE master encryption key.

   ```
   SQL> ADMINISTER KEY MANAGEMENT CREATE KEY FORCE KEYSTORE IDENTIFIED BY
   <hsm_partition_password>;
   ```

4. List all the key IDs.

   ```
   SQL> SELECT KEY_ID FROM V$ENCRYPTION_KEYS;
   ```

To activate a TDE Master Encryption Key

1. Log in to the database instance as a user who has been granted the **ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege.

   ```
   SQL> connect system/<password>
   ```

2. Open the keystore if it is not opened.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
   <hsm_partition_password>;
   ```

3. Query the **KEY_ID** column of the **V$ENCRYPTION_KEYS** view to find the key identifier.

   ```
   SQL> SELECT KEY_ID FROM V$ENCRYPTION_KEYS;
   ```

4. Use this key identifier to activate the TDE master encryption key.

   ```
   SQL> ADMINISTER KEY MANAGEMENT USE KEY
   'key_identifier_from_V$ENCRYPTION_KEYS' IDENTIFIED BY
   <hsm_partition_password>;
   ```

5. You can find the key id that is in use by issuing the below command:

   ```
   SQL> SELECT KEY_ID FROM V$ENCRYPTION_KEYS WHERE ACTIVATION_TIME = (SELECT
   MAX(ACTIVATION_TIME) FROM V$ENCRYPTION_KEYS WHERE ACTIVATING_DBID = (SELECT
   DBID FROM V$DATABASE));
   ```

Encrypting an Existing Tablespaces with Online Conversion

You can encrypt an existing data file of a user tablespace when the tablespace is online.

To encrypt an existing user-defined tablespace with online conversion

1. Log in to the database instance as a user who has been granted the **ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege.

   ```
   SQL> connect system/<password>
   ```

2. Open the Hardware Keystore.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
   <hsm_partition_password>;
   ```

3. After you have opened the hardware keystore, you are ready to set the hardware keystore TDE master encryption key.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEY FORCE KEYSTORE IDENTIFIED BY
   <hsm_partition_password>;
   ```

4. Create any tablespace for demonstration.

   ```
   SQL> CREATE TABLESPACE TESTTBSPACE DATAFILE
   '/u01/app/oracle/oradata/CDB1/UNSECURE01.DBF' SIZE 150M;
   ```

5. Create a table.

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

6. Insert some values into the table.

   ```
   SQL> INSERT INTO EMPLOYEE VALUES (001,'JOHN SMITH',15000);

   SQL> INSERT INTO EMPLOYEE VALUES (002,'SCOTT TIGER',25000);

   SQL> INSERT INTO EMPLOYEE VALUES (003,'DIANA HAYDEN',35000);
   ```

7. Ensure that the **COMPATIBLE** initialization parameter is set correctly according to the database version.

   ```
   SQL> SHOW PARAMETER COMPATIBLE
   ```

8. Encrypt the tablespace.

   ```
   SQL> ALTER TABLESPACE TESTTBSPACE ENCRYPTION ONLINE USING 'AES192' ENCRYPT
   FILE_NAME_CONVERT = ('UNSECURE01.DBF','SECURE01.DBF');
   ```

   NOTE: For Oracle Database 12c and 18c you must use the FILE_NAME_CONVERT clause for non-Oracle managed files. For Oracle Database 19c, if you omit the FILE_NAME_CONVERT clause, then Oracle Database internally assigns an auxiliary file name and then later renames it back to the original name.

   NOTE: In an Oracle-managed files configuration, new data files are created automatically.

9. Fetch data from table.

```
SQL> Select * from employee;
```

10. Close wallet.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE close IDENTIFIED BY
<hsm_partition_password>;
```

11. Fetch data from table.

```
SQL> Select * from employee;
```

Nothing is displayed, as the wallet is closed.

Decrypting an Existing Tablespace with Online Conversion

To decrypt an existing tablespace with online conversion, you can use the **ALTER TABLESPACE SQL** statement with **DECRYPT** clause.

To decrypt an existing tablespace with online conversion

1. List all the tablespaces that are encrypted.

```
SQL> SELECT TABLESPACE_NAME, ENCRYPTED FROM DBA_TABLESPACES;
```

2. As a user who has been granted the **ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege, open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<hsm_partition_password>;
```

3. Ensure that the **COMPATIBLE** initialization parameter is set correctly according to the database version.

```
SQL> SHOW PARAMETER COMPATIBLE
```

4. Run the ALTER TABLESPACE SQL statement with the DECRYPT clause.

```
SQL> ALTER TABLESPACE TESTTBSPACE ENCRYPTION ONLINE DECRYPT
FILE_NAME_CONVERT = ('SECURE01.DBF','UNSECURE01.DBF');
```

> **NOTE:** For Oracle Database 12c and 18c, you must use the FILE_NAME_CONVERT clause for non-Oracle managed files. For Oracle Database 19c, if you omit the FILE_NAME_CONVERT clause, then Oracle Database internally assigns an auxiliary file name and later renames it back to the original name.

> **NOTE:** In an Oracle-managed files configuration, new data files are created automatically.

5. Verify that the tablespace is no longer encrypted.

```
SQL> SELECT TABLESPACE_NAME, ENCRYPTED FROM DBA_TABLESPACES;
```

Rekeying an Existing Tablespace with Online Conversion

To rekey an existing tablespace that is online, you can use the **REKEY** clause of the **ALTER TABLESPACE** SQL statement.

To rekey an existing tablespace with online conversion

1. To find the current status of the encryption algorithm used by the master encryption key, run the following sql query.

   ```
   SQL> SELECT * FROM V$ENCRYPTED_TABLESPACES;
   ```

   NOTE: Do not perform an online tablespace rekey operation with a master key operation concurrently.

2. Ensure that the **COMPATIBLE** initialization parameter is set correctly according to the database version.

   ```
   SQL> SHOW PARAMETER COMPATIBLE
   ```

3. Perform the rekey operation if the key version status of the tablespace is **NORMAL**.

   ```
   SQL> ALTER TABLESPACE UNSECURESPACE ENCRYPTION USING 'AES192' REKEY
   FILE_NAME_CONVERT = ('SECURE01.DBF', 'SECURE02.DBF');
   ```

   NOTE: For Oracle Database 12c and 18c you must use the FILE_NAME_CONVERT clause for non-Oracle managed files. For Oracle Database 19c if you omit the FILE_NAME_CONVERT clause, then Oracle Database internally assigns an auxiliary file name, and then later renames it back to the original name.

   NOTE: In an Oracle-managed files configuration, new data files are created automatically.

4. Verify that rekey is successful.

   ```
   SQL> SELECT * FROM V$ENCRYPTED_TABLESPACES;
   ```

Encrypting an Existing User-Defined Tablespace with Offline Conversion

You can encrypt an existing data file of a user tablespace when the tablespace is offline

To Encrypt an existing user-defined tablespace with offline conversion

5. Log in to the database instance as a user who has been granted the ADMINISTER

   KEY MANAGEMENT or SYSKM privilege.

   ```
   SQL> connect system/<password>
   ```

6. Open the Hardware Keystore.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
   <hsm_partition_password>;
   ```

7. After you have opened the hardware keystore, you are ready to set the hardware keystore TDE master encryption key.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEY FORCE KEYSTORE IDENTIFIED BY
   <hsm_partition_password>;
   ```

8. Create any tablespace for demonstration.

```
SQL> CREATE TABLESPACE TESTTBSPACE DATAFILE
'/u01/app/oracle/oradata/CDB1/UNSECURE01.DBF' SIZE 150M;
```

9. Create a table.

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

10. Insert some values into the table.

```
SQL> INSERT INTO EMPLOYEE VALUES (001,'JOHN SMITH',15000);
SQL> INSERT INTO EMPLOYEE VALUES (002,'SCOTT TIGER',25000);
SQL> INSERT INTO EMPLOYEE VALUES (003,'DIANA HAYDEN',35000);
```

11. Bring the tablespace offline.

```
SQL> ALTER TABLESPACE TESTTBSPACE OFFLINE NORMAL;
```

> **NOTE:** The offline conversion method does not use auxiliary disk space or files, and it operates directly in-place to the data files. Therefore, you should perform a full backup of the user tablespace before converting it offline.

12. Encrypt the tablespace.

```
SQL> ALTER TABLESPACE TESTTBSPACE ENCRYPTION OFFLINE ENCRYPT;
```

To encrypt individual data files within a tablespace, run the following command:

```
SQL> ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/CDB1/UNSECURE01.DBF'
ENCRYPT;
```

13. Bring the tablespace back online.

```
SQL> ALTER TABLESPACE TESTTBSPACE ONLINE;
```

14. Fetch data from table.

```
SQL> Select * from employee;
```

15. Close wallet.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE close IDENTIFIED BY
<hsm_partition_password>;
```

16. Fetch data from table.

```
SQL> Select * from employee;
```

Nothing is displayed as the wallet is closed.

Decrypting an Existing Tablespace with Offline Conversion

To decrypt an existing tablespace with offline conversion, you can use the **ALTER TABLESPACE SQL** statement with the **OFFLINE** and **DECRYPT** clauses.

To decrypt an Existing User-Defined Tablespace with Offline Conversion

1. List all the tables that are encrypted.

```
SQL> SELECT TABLESPACE_NAME, ENCRYPTED FROM DBA_TABLESPACES;
```

2. Bring the tablespace offline.

```
SQL> ALTER TABLESPACE TESTTBSPACE OFFLINE NORMAL;
```

3. As a user who has been granted the **ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege, open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<hsm_partition_password>;
```

4. Run the **ALTER TABLESPACE SQL** statement to perform the decryption.

```
SQL> ALTER TABLESPACE TESTTBSPACE ENCRYPTION OFFLINE DECRYPT;
```

5. Bring the tablespace online.

```
SQL> ALTER TABLESPACE TESTTBSPACE ONLINE;
```

Migrating Master Encryption Key from software keystore to hardware keystore

The following procedures detail how to migrate an existing Master Encryption Key from software keystore to hardware keystore.

> **NOTE:** It is assumed that no software-based wallet is yet created in the directory you would specify to create one.

To verify that the software-based wallet is working

1. Refer to Appendix B and complete the software keystore configuration based on the installed Oracle Database version.

> Configure Software Keystore for Oracle Database 12c

> Configure Software Keystore for Oracle Database 18c and 19c

2. Start the database.

```
$ sqlplus / as sysdba
```

If the database is not yet started, you can start it using:

```
SQL> startup;
```

3. Grant the **ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege to the **SYSTEM** and any additional user that you want to grant privilege to.

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

```
SQL> commit;
```

4. Connect to the database as **'system'**.

```
SQL> connect system/<password>
```

> **NOTE:** Password for 'system' is set during Oracle installation. All dbapasswords throughout this document are set to "temp123#".

5. Run the ADMINISTER KEY MANAGEMENT SQL statement to create the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY
<software_keystore_password>;
```

NOTE: For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'keystore_location' IDENTIFIED BY <software_keystore_password>;** where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.

6. Run the **ADMINISTER KEY MANAGEMENT SQL** statement to open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<software_keystore_password>;
```

7. Set the master encryption key in the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY
<software_keystore_password> WITH BACKUP USING 'backup_identifier';
```

NOTE: Including the parameter WITH BACKUP creates a backup of the keystore. You must use this option for password based keystores. You can optionally use the USING clause to add a brief description of the backup. Enclose this description in single quotation marks (' '). This identifier is appended to the named keystore file (for example, ewallet_time_stamp_emp_key_backup.p12, with emp_key_backup being the backup identifier).

To verify the master encryption key is encrypting the Oracle database

1. Create a CUSTOMERS table in the database.

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

2. Enter some values in the CUSTOMERS table.

```
SQL> INSERT INTO CUSTOMERS VALUES (001, 'George Bailey', 10000);

SQL> INSERT INTO CUSTOMERS VALUES (002, 'Denial Vettory', 20000);

SQL> INSERT INTO CUSTOMERS VALUES (003, 'MS Dhoni', 30000);

SQL> INSERT INTO CUSTOMERS VALUES (004, 'Shahid Afridi', 40000);
```

3. Encrypt the 'CREDIT_LIMIT' column of the 'CUSTOMERS' table.

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

4. With the next command, the values listed in the encrypted column are returned in clear text; Transparent Data Encryption decrypts them automatically.

```
SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;
```

5. List encrypted columns in your database.

```
SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;
```

6. View information about the software keystore.

```
SQL> SELECT * FROM V$ENCRYPTION_WALLET;
```

7. Create an encrypted tablespace.

```
SQL> CREATE TABLESPACE SECURESPACE DATAFILE
'/u01/app/oracle/oradata/orcl/SECURE01.DBF' SIZE 150M ENCRYPTION DEFAULT
STORAGE (ENCRYPT);
```

8. Create a table in the tablespace.

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

9. Insert some values in EMPLOYEE table.

```
SQL> INSERT INTO EMPLOYEE VALUES (001,'JOHN SMITH',15000);

SQL> INSERT INTO EMPLOYEE VALUES (002,'SCOTT TIGER',25000);

SQL> INSERT INTO EMPLOYEE VALUES (003,'DIANA HAYDEN',35000);
```

10. Display the contents of the EMPLOYEE table with the following command:

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

11. Close the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<software_keystore_password>;
```

12. After closing the keystore execute the following command to display the contents again:

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

You will get the following error that means you cannot list the contents of EMPLOYEE table, if the keystore is closed.

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

13. Open the keystore:

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<software_keystore_password>;

SQL> exit
```

To test if the database can access the HSM

1. Refer to Appendix B and complete the Keystore for Migration from Software to Hardware configuration based on the installed Oracle Database version.

> Configure Keystore for Migration from Software to Hardware for Oracle Database 12c

> Configure Keystore for Migration from Software to Hardware for Oracle Database 18c and 19c

2. Connect to the database as 'system'.

```
SQL> connect system/<password>
```

3. Migrate the wallet onto the HSM device.

```
SQL> ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY
"hsm_partition_pwd" MIGRATE USING <software_keystore_password> WITH BACKUP
USING 'backup_identifier';
```

NOTE: "hsm_partition_pwd" is the password for the HSM partition. The 'MIGRATE USING software_keystore' string re-encrypts the Transparent Data Encryption column

keys and tablespace keys with the new HSM based master key. The <software_keystore_password> is the password of software wallet in the step 1.

NOTE: In case a network latency is observed and heartbeat check fails with DPoD, Refer to Troubleshooting Tips Problem 4.

4. Return the values in the encrypted column to clear text; Transparent Data Encryption decrypts them automatically, now using the HSM master key.

```
SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;
```

5. Change the password of software keystore to be the same as HSM partition password. Ensure that the software wallet is open.

```
SQL> ADMINISTER KEY MANAGEMENT ALTER KEYSTORE PASSWORD IDENTIFIED BY
<software_keystore_password> SET hsm_partition_pwd WITH BACKUP USING
'backup_identifier';
```

   Now, when you open the keystore, it will open both the software-based keystore and the HSM-based keystore.

6. Close the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
"hsm_partition_pwd";
```

7. Open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
"hsm_partition_pwd";
```

   This opens both the HSM and the software keystore.

8. Check the wallet information with the following command.

```
SQL> SELECT * FROM V$ENCRYPTION_WALLET;
```

9. Change the password back to the initial password for the software based wallet (if you want to) and use the following syntax to create an auto-login keystore for the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
IDENTIFIED BY <software_keystore_password>;
```

NOTE: For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE 'keystore_location' IDENTIFIED BY <software_keystore_password>;** where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.

NOTE: To use the auto-login wallet only on local system include LOCAL AUTO_LOGIN instead of AUTO_LOGIN.

10. Verify that an auto-open software keystore has been created in the oracle wallet directory you specified in the **sqlnet.ora** file or **wallet_root** parameter: You will find two wallets in this directory: "**ewallet.p12**"

and "**cwallet.sso**"; the latter is the auto-open wallet. Move or rename the encryption wallet "**ewallet.p12**" to ensure that Oracle uses the auto-open wallet.

```
# mv ewallet.p12 ewallet.p24
```

11. Restart the database and connect to the database as system. Open the HSM keystore, the software wallet will open automatically.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
"hsm_partition_pwd";
```

Configuring Auto-login for Hardware Keystore

To create the HSM Auto Wallet

> **NOTE:** It is assumed that no software wallet has been created so far. If a software wallet or auto wallet has already been created, you need to skip step 3 below and remove/rename the cwallet.sso file.

1. Close the Hardware Keystore if it is opened.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<hsm_partition_password>;
```

2. Refer to Appendix B and complete the Software keystore configuration based on the installed Oracle Database version:

> Configure Software Keystore for Oracle Database 12c

> Configure Software Keystore for Oracle Database 18c and 19c

3. Create the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY
<software_keystore_password>;
```

> **NOTE:** For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'keystore_location' IDENTIFIED BY <software_keystore_password>;** where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.

> **NOTE:** If a software wallet has already been created, you need to skip this step and remove/rename the cwallet sso file.

4. Open the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<software_keystore_password>;
```

5. Add the HSM password as a client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'hsm_partition_password' FOR
CLIENT 'HSM_PASSWORD' IDENTIFIED BY <software_keystore_password> WITH
BACKUP USING 'backup_identifier';
```

6. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<software_keystore_password>;
```

7. Create Auto-Login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
IDENTIFIED BY <software_keystore_password>;
```

> **NOTE:** For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE 'keystore_location' IDENTIFIED BY <software_keystore_password>;** where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.

8. Refer to Appendix B and complete the Keystore for Migration from Software to Hardware configuration based on the installed Oracle Database version:

> Configure Keystore for Migration from Software to Hardware for Oracle Database 12c

> Configure Keystore for Migration from Software to Hardware for Oracle Database 18c and 19c

   At this stage, close the database and open it one more time and the next time when a TDE operation executes, the hardware security module auto-login keystore opens automatically.

9. Restart the database and connect as a system.

10. Check the wallet information with the following command

```
SQL> SELECT * FROM V$ENCRYPTION_WALLET;
```

Working with Pluggable Databases (PDB)

Oracle multitenant architecture enables an Oracle database to function as a multitenant container database (CDB) that includes zero, one, or many customer-created pluggable databases (PDBs). A PDB is a portable collection of schemas, schema objects, and non-schema objects that appears to an Oracle Net client as a non-CDB. All Oracle databases before Oracle Database 12c were non-CDBs.

TDE in Pluggable Databases

You can use TDE in PDBs as you would in Oracle database configurations.

To use TDE with Pluggable Databases:

1. Edit the tnsnames.ora file to add a new service for the PDB. By default, the tnsnames.ora file is located in the "`ORACLE_HOME/network/admin`" directory or in the location set by the TNS_ADMIN environment variable. Ensure that you have properly set the TNS_ADMIN environment variable to point to the correct `tnsnames.ora` file.

   For example:

```
salespdb =
(DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
```

```
    (SERVICE_NAME = salespdb.localdomain)
  )
    )
```

Where, **salespdb** is the new Pluggable database name.

2. Restart the Listener Service.

```
$ lsnrctl stop
$ lsnrctl start
```

For RAC:

```
$ srvctl stop listener -listener  LISTENER
$ srvctl start listener -listener  LISTENER
```

3. Refer to Appendix B and complete the hardware keystore configuration based on the installed Oracle Database version.

> Configure Hardware Keystore for Oracle Database 12c

> Configure Hardware Keystore for Oracle Database 18c and 19c

4. Start the **sqlplus** session to connect to PDB.

```
$ sqlplus / as sysdba
```

5. Log in to the database instance as a user who has been granted **the ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege.

```
SQL> connect system/<password>
```

6. Open the hardware keystore in the **CDB$ROOT** container.

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

7. Set the master encryption key in the **CDB$ROOT** container onto HSM. Skip this step if the master encryption key is already generated onto HSM.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY FORCE KEYSTORE IDENTIFIED BY
<hsm_partition_password>;
```

8. Connect as sysdba.

```
SQL> connect / as sysdba
```

9. Open the pdb in read write mode.

```
SQL> ALTER PLUGGABLE DATABASE <PDB_NAME> OPEN READ WRITE;
```

> **NOTE:** In a RAC environment pluggable database must be open on all RAC nodes.

10. Set the container to the pdb.

```
SQL> ALTER SESSION SET CONTAINER=<pdb_name>;
```

Or execute the below command:

```
SQL> Connect system/<system_password>@Pluggable Database Service name
```

For Example:

```
SQL> connect system/temp123#@salespdb
```

11. Grant the following privileges to the PDB Admin:

```
SQL> GRANT ADMINISTER KEY MANAGEMENT TO salesadm;
SQL> GRANT CREATE SESSION TO salesadm;
SQL> GRANT CONNECT TO salesadm;
SQL> GRANT DBA TO salesadm;
SQL> GRANT CREATE ANY TABLE TO salesadm;
SQL> GRANT UNLIMITED TABLESPACE TO salesadm;
SQL> ALTER USER salesadm PROFILE DEFAULT;
SQL> COMMIT;
```

Where, **salesadm** is the administrative user name created at the time of creating PDB.

12. Connect to the PDB using the PDB username. If you are able to connect, the configuration was completed successfully.

```
SQL> Connect pdbuser/<system_password>@Pluggable Database Service name
```

For Example:

```
SQL> connect salesadm/temp123#@salespdb
```

To generate a TDE master encryption key for PDB

1. Start the **sqlplus** session to connect to PDB.

```
$ sqlplus / as sysdba
SQL> Connect <pdb_admin>/<pdb_admin_password>@Pluggable Database Service
name
```

For example:

```
SQL> connect salesadm/temp123#@salespdb
```

2. Run the **ADMINISTER KEY MANAGEMENT** SQL statement using the following syntax:

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

> **NOTE:** Before opening the keystore and generating the Master key for PDB, ensure that the keystore for CDB (root container) is opened and the Master key for CDB is generated. Do not configure the HSM auto login for CDB until you generate the master key for PDB (this applies to all PDBs that are using the TDE). After generating the Master key for all PDBs you can configure the CDB for auto login, and it will work for all PDBs.

3. Create the PDB Master Key.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY
"<hsm_partition_password>";
```

> **NOTE:** This will generate a new master key, any encryption/decryption operations performed within this PDB will use this master key.
>
> **NOTE:** In case a network latency is observed and heartbeat check fails with DPoD, Refer to Troubleshooting Tips Problem 4.

To verify the master encryption key is encrypting the PDB

1. Create a CUSTOMERS table in the PDB.

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

2. Enter some values in the CUSTOMERS table.

   ```
   SQL> INSERT INTO CUSTOMERS VALUES (001, 'George Bailey', 10000);
   ```

   ```
   SQL> INSERT INTO CUSTOMERS VALUES (002, 'Denial Vettory', 20000);
   ```

3. Encrypt the 'CREDIT_LIMIT' column of the 'CUSTOMERS' table.

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

4. List the values in the encrypted column. Transparent Data Encryption decrypts them automatically and the values are returned in clear text.

   ```
   SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;
   ```

5. List encrypted columns in your databases.

   ```
   SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;
   ```

6. Create an encrypted tablespace.

   ```
   SQL> CREATE TABLESPACE SECURESPACE DATAFILE
   '/u01/app/oracle/oradata/orcl/salespdb/SECURE01.DBF' SIZE 150M ENCRYPTION
   DEFAULT STORAGE (ENCRYPT);
   ```

7. Create a table in the tablespace.

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

8. Insert some values in EMPLOYEE table.

   ```
   SQL> INSERT INTO EMPLOYEE VALUES (001,'JOHN SMITH',15000);
   ```

   ```
   SQL> INSERT INTO EMPLOYEE VALUES (002,'SCOTT TIGER',25000);
   ```

   ```
   SQL> INSERT INTO EMPLOYEE VALUES (003,'DIANA HAYDEN',35000);
   ```

9. Display the contents of the EMPLOYEE table with the following command.

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

10. Close the keystore.

    ```
    SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
    "hsm_partition_password";
    ```

11. After closing the keystore execute the command to display the contents again.

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

You will get the following error that means you cannot list the contents of EMPLOYEE table, if keystore is closed.

```
ERROR at line 1:

ORA-28365: wallet is not open
```

12. Open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
"hsm_partition_password";
```

Unplugging and Plugging a PDB with Encrypted Data in a CDB in United Mode

You can unplug a PDB from one CDB that has been configured with an HSM and then plug it into another CDB also configured with a hardware keystore.

> **NOTE:** This feature has been tested with Oracle Database 18c and 19c only.

To unplug a pdb

1. Connect as sysdba.

```
SQL> connect / as sysdba
```

2. Check if the PDB is unplugged or not.

```
SQL> select status,pdb_name from dba_pdbs;
```

3. Close the pdb before unplugging it.

```
SQL> ALTER PLUGGABLE DATABASE PDB1 CLOSE IMMEDIATE;
```

4. Unplug the PDB.

```
SQL> ALTER PLUGGABLE DATABASE PDB1 UNPLUG INTo
'/u01/app/oracle/oradata/pdb1.xml';
```

5. Verify that the PDB has been unplugged.

```
SQL> select STATUS from dba_pdbs;
```

> **NOTE:** Ensure to move the master encryption keys of the unplugged PDB in the hardware keystore that was used at the source CDB to the hardware keystore that is in use at the destination. Follow the *Thales Luna HSM Product Documentation* for detailed steps for moving the keys if you are using a different HSM on the destination.

To plug a PDB into another CDB

1. Ensure that master key is set and open in root container of destination database using Hardware Keystore.
2. List the PDBs.

```
SQL> show pdbs
```

3. Plug the unplugged PDB into the destination CDB that has been configured with the hardware keystore.

```
SQL> create pluggable database pdbfromcdb1 using
'/u01/app/oracle/oradata/pdb1.xml'
file_name_convert=('/u01/app/oracle/oradata/ORCL/pdb1','/u01/app/oracle/ora
data/CDB2/pdbfromcdb1') KEYSTORE IDENTIFIED BY <hsm_partition_password>;
```

4. You can check if a PDB has already been plugged in by querying the STATUS column of the DBA_PDBS data dictionary view.

```
SQL> select STATUS from dba_pdbs;
```

5. Open the pluggable database in **read-write** mode.

```
SQL> alter pluggable database PDB1_FROMCDB1 open;
```

6. Log in to the plugged PDB as the same user in the pdb who was granted the **ADMINISTER KEY MANAGEMENT** or **SYSKM** privilege.
7. Open the master encryption key of the plugged PDB.

```
SQL> ALTER SESSION SET CONTAINER = PDB1_FROMCDB1;
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<hsm_partition_password>;
```

8. Import the hardware keystore master encryption key into the PDB.

```
SQL> ADMINISTER KEY MANAGEMENT IMPORT ENCRYPTION KEYS WITH SECRET "HSM"
FROM 'HSM' IDENTIFIED BY <hsm_partition_password>;
```

9. Restart the PDB.

```
SQL> ALTER PLUGGABLE DATABASE PDB1_FROMCDB1 close;
```

```
SQL> ALTER PLUGGABLE DATABASE PDB1_FROMCDB1 open;
```

Storing Oracle Database secrets in a hardware keystore

Secrets are data that support internal Oracle Database features that integrate external clients, such as Oracle GoldenGate, into the database.

To store and update Oracle Database secrets

1. Log in to the database instance as a user who has been granted **the ADMINISTER**

   KEY MANAGEMENT or **SYSKM** privilege.

```
SQL> connect system/<password>
```

2. Open the Hardware Keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<hsm_partition_password>;
```

> **NOTE:** Ensure that the Master Encryption Key is generated on the HSM before creating or updating a secret.

> To add the secret:

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'mysecret' FOR CLIENT 'test'
USING TAG 'testsecret' IDENTIFIED BY temp123#;
```

> To update the secret:

```
SQL> ADMINISTER KEY MANAGEMENT UPDATE SECRET 'mynewsecret' FOR CLIENT
'test' USING TAG 'testsecret' IDENTIFIED BY temp123#;
```

> To delete the secret:

```
SQL> ADMINISTER KEY MANAGEMENT DELETE SECRET FOR CLIENT 'test' IDENTIFIED
BY temp123#;
```

CHAPTER 3: Integrating Luna HSM with Oracle Database RAC

This chapter covers the following topics:

Oracle Database RAC Setup

We recommend you familiarize yourself with the Oracle Database RAC documentation. We recommend reading the *Oracle Database RAC documentation* for more information about installing and pre-installation requirements.

The three machines in this configuration are given the following example names:

- RAC1.localdomain
- RAC2.localdomain
- RAC3.localdomain

This demonstration uses three Oracle Homes. Each Oracle Home has one database and three instances for every database. The setup has 3x3 Oracle RAC setup. You can scale the setup as per your requirement.

Supported Platforms

The following platforms are supported for Luna HSM:

Oracle Database 19c

Luna HSM	Platforms Tested
Luna HSM	Oracle Enterprise Linux

Oracle Database 18c

Luna HSM	Platforms Tested
Luna HSM	Red Hat Enterprise Linux

Oracle Database 12c

Luna HSM	Platforms Tested
Luna HSM	Red Hat Enterprise Linux AIX Solaris SPARC

Verifying the Oracle RAC installation

Before proceeding for HSM based wallet management, it is assumed that Oracle RAC is setup properly and running at this point. You can verify the RAC running information by executing the following commands on any RAC instance:

```
# crsctl stat res -t
--------------------------------------------------------------------------------
Name           Target  State        Server                   State details
--------------------------------------------------------------------------------
Local Resources
--------------------------------------------------------------------------------
ora.DATA.dg
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
               ONLINE  ONLINE       rac3                     STABLE
ora.LISTENER.lsnr
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
               ONLINE  ONLINE       rac3                     STABLE
ora.asm
               ONLINE  ONLINE       rac1                     Started,STABLE
               ONLINE  ONLINE       rac2                     Started,STABLE
               ONLINE  ONLINE       rac3                     Started,STABLE
```

```
ora.net1.network
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
               ONLINE  ONLINE       rac3                     STABLE
ora.ons
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
               ONLINE  ONLINE       rac3                     STABLE
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
ora.LISTENER_SCAN1.lsnr
      1        ONLINE  ONLINE       rac2                     STABLE
ora.LISTENER_SCAN2.lsnr
      1        ONLINE  ONLINE       rac1                     STABLE
ora.LISTENER_SCAN3.lsnr
      1        ONLINE  ONLINE       rac3                     STABLE
ora.MGMTLSNR
      1        ONLINE  ONLINE       rac2                     169.254.110.224
192.
                                                             168.1.102,STABLE
ora.cvu
      1        ONLINE  ONLINE       rac3                     STABLE
ora.mgmtdb
      1        ONLINE  ONLINE       rac2                     Open,STABLE
ora.oc4j
      1        ONLINE  ONLINE       rac3                     STABLE
ora.orcl1rac.db
      1        ONLINE  ONLINE       rac1                     Open,STABLE
      2        ONLINE  ONLINE       rac2                     Open,STABLE
      3        ONLINE  ONLINE       rac3                     Open,STABLE
ora.orcl2rac.db
      1        ONLINE  ONLINE       rac1                     Open,STABLE
      2        ONLINE  ONLINE       rac2                     Open,STABLE
```

```
      3        ONLINE  ONLINE       rac3                     Open,STABLE
ora.orcl3rac.db
      1        ONLINE  ONLINE       rac1                     Open,STABLE
      2        ONLINE  ONLINE       rac2                     Open,STABLE
      3        ONLINE  ONLINE       rac3                     Open,STABLE
ora.rac1.vip
      1        ONLINE  ONLINE       rac1                     STABLE
ora.rac2.vip
      1        ONLINE  ONLINE       rac2                     STABLE
ora.rac3.vip
      1        ONLINE  ONLINE       rac3                     STABLE
ora.scan1.vip
      1        ONLINE  ONLINE       rac2                     STABLE
ora.scan2.vip
      1        ONLINE  ONLINE       rac1                     STABLE
ora.scan3.vip
      1        ONLINE  ONLINE       rac3                     STABLE
--------------------------------------------------------------------------------
```

The setup has three databases: ORCL1RAC, ORCL2RAC, and ORCL3RAC. Each database has three instances which run simultaneously on three nodes. Below are the details of setup:

DATABASE ORCL1RAC

```
[oracle@rac1 ~]$ srvctl config database -d ORCL1RAC
Database unique name: orcl1rac
Database name: orcl1rac
Oracle home: /u01/app/oracle/product/12.1.0.2/db_1
Oracle user: oracle
Spfile: +DATA/ORCL1RAC/PARAMETERFILE/spfile.301.910221979
Password file: +DATA/ORCL1RAC/PASSWORD/pwdorcl1rac.276.910221645
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
```

```
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: orcl1rac1,orcl1rac2,orcl1rac3
Configured nodes: rac1,rac2,rac3
Database is administrator managed

[oracle@rac1 ~]$ srvctl status database -d ORCL1RAC
Instance orcl1rac1 is running on node rac1
Instance orcl1rac2 is running on node rac2
Instance orcl1rac3 is running on node rac3
```

DATABASE ORCL2RAC

```
[oracle@rac2 ~]$ srvctl config database -d ORCL2RAC
Database unique name: orcl2rac
Database name: orcl2rac
Oracle home: /u01/app/oracle/product/12.1.0.2/db_2
Oracle user: oracle
Spfile: +DATA/ORCL2RAC/PARAMETERFILE/spfile.331.910223671
Password file: +DATA/ORCL2RAC/PASSWORD/pwdorcl2rac.306.910223319
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups: DATA
Mount point paths:
Services:
```

```
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: orcl2rac1,orcl2rac2,orcl2rac3
Configured nodes: rac1,rac2,rac3
Database is administrator managed
[oracle@rac2 ~]$ srvctl status database -d ORCL2RAC
Instance orcl2rac1 is running on node rac1
Instance orcl2rac2 is running on node rac2
Instance orcl2rac3 is running on node rac3
```

DATABASE ORCL3RAC

```
[oracle@rac3 ~]$ srvctl config database -d ORCL3RAC
Database unique name: orcl3rac
Database name: orcl3rac
Oracle home: /u01/app/oracle/product/12.1.0.2/db_3
Oracle user: oracle
Spfile: +DATA/ORCL3RAC/PARAMETERFILE/spfile.361.910224445
Password file: +DATA/ORCL3RAC/PASSWORD/pwdorcl3rac.336.910224083
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
```

```
OSOPER group: oper
Database instances: orcl3rac1,orcl3rac2,orcl3rac3
Configured nodes: rac1,rac2,rac3
Database is administrator managed

[oracle@rac3 ~]$ srvctl status database -d ORCL3RAC
Instance orcl3rac1 is running on node rac1
Instance orcl3rac2 is running on node rac2
Instance orcl3rac3 is running on node rac3
The V$ACTIVE_INSTANCES view can also display the current status of the
instances.
--------------------------------------------------------------------------------
$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.2.0 Production on Wed Apr 27 20:31:35 2016

Copyright (c) 1982, 2014, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage
Management, OLAP,
Advanced Analytics and Real Application Testing options
SQL> SELECT inst_name FROM v$active_instances;

INST_NAME
---------------------------------------------------------
rac1.localdomain:orcl1rac1
rac2.localdomain:orcl1rac2
rac3.localdomain:orcl1rac3
---------------------------------------------------------
```

The above details demonstrate that the Oracle RAC 3x3 node setup is working as expected. Now setup Luna HSM for Oracle TDE.

Setting up Luna HSM for Transparent Data Encryption with Oracle RAC

To start using HSM-based encryption, you need to have a master encryption key that will be stored inside the HSM. Use the master encryption key to encrypt or decrypt column encryption keys inside the HSM. The HSM can be used in the following ways to protect the master encryption key:

> **NOTE:** The example setup shows three Oracle Homes. Each demonstrate one of the above described scenarios.

Configuring the PKCS11 Provider on Oracle RAC Instances

To set up Luna HSM for TDE with Oracle RAC, perform the following steps on RAC1, RAC2 and RAC3:

Oracle requires access to the PKSC#11 library provided by the HSM. Copy the Luna HSM PKCS#11 library to the specified directory structure recommended by Oracle. Use the following directory structures.

Linux/Solaris/ AIX	`"/opt/oracle/extapi/[32,64]/hsm/{Vendor}/{Version}/libapiname.ext"`
Windows	`"%SYSTEMDRIVE%\oracle\extapi\[32,64]\hsm\{Vendor}\{Version}\libapiname.ext"`

Parameter	Definition
[32,64]	Specifies whether the supplied binary is 32-bits or 64-bits.
[vendor]	Specifies the name of the vendor suppling the library.
[version]	Refers to the version number of the library. **Note:** The version number should be in the format [number.number.number]. The API name requires no special format. The XX must be prefixed with the word lib.
.ext	The extension must be replaced by the extension of the library file.

Only one PKCS#11 library is supported at a time. Oracle user should have the read/write permission of the above directory.

For example,

`"/opt/oracle/extapi/64/hsm/safenet/7.x.x/libCryptoki2_64.so"` (UNIX)

Below are the commands to create the directory and setup the SafeNet library.

```
# mkdir -p /opt/oracle/extapi/64/hsm/safenet/7.x.x
```

```
# cp /usr/safenet/lunaclient/lib/libCryptoki2_64.so
/opt/oracle/extapi/64/hsm/safenet/7.x.x/
```

```
# chown -R oracle:oinstall /opt/oracle/
# chmod -R 775 /opt/oracle/
```

Migrating Master Encryption Key from software wallet to HSM

The following procedures detail how to migrate an existing Master Encryption Key for HSM-based encryption onto an HSM device.

> **NOTE:** The following step is needed if you are not using the shared disk for storing the wallet. Storing the wallet on shared disk will ease the process of copying the wallet manually on all instances. You can use a NAS/NFS server for shared storage or ASMCA utility to create the ACFS (ASM Cluster File Systems) file and mount this file on a disk that will be used by all instances. For more information, see the Oracle Documentation on Creating ACFS file for Storing Wallet on Clustered System.
>
> It is assumed that no software-based wallet is yet created in the directory you would specify to create one.

Create the directory for every database and permit the oracle user to access this directory on RAC1, RAC2, and RAC3 instances. Run the following commands on each RAC instance:

```
# mkdir -pv /etc/oracle/wallet/ORCL1RAC
# mkdir -pv /etc/oracle/wallet/ORCL2RAC
# mkdir -pv /etc/oracle/wallet/ORCL3RAC
# cd /etc
# chown -R oracle:oinstall oracle/wallet/
# chmod -R 775 oracle/wallet/
```

Verify that the 'traditional' software-based wallet is working properly

1. Refer to Appendix B and complete the Software keystore configuration on all nodes, based on the installed Oracle Database version.

> Configure Software Keystore for Oracle Database 12c

> Configure Software Keystore for Oracle Database 18c and 19c

2. Start the database.

   ```
   $ sqlplus / as sysdba
   ```

   If the database is not yet started, restart the database.

   ```
   SQL> startup;
   ```

3. Grant ADMINISTER KEY MANAGEMENT or SYSKM privilege to SYSTEM and any user that you want to use.

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

4. Connect to the database as 'system'.

   ```
   SQL> connect system/<password>
   ```

NOTE: Password for 'system' can be set during Oracle installation. All dbapasswords throughout this document has been set to "Temp1234".

5. Run the ADMINISTER KEY MANAGEMENT SQL statement to create the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY
<software_keystore_password>;
```

NOTE: For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'keystore_location' IDENTIFIED BY <software_keystore_password>;** where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.

6. Run the ADMINISTER KEY MANAGEMENT SQL statement to open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<software_keystore_password>;
```

Repeat steps1-6, on all instances (RAC1, RAC2, and RAC3). If you are storing the wallet on a shared disk, only repeat steps 1-4.

7. Set the master encryption key in the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY
<software_keystore_password> WITH BACKUP USING 'backup_identifier';
```

NOTE: WITH BACKUP creates a backup of the keystore. You must use this option for password-based keystores. You can optionally use the USING clause to add a brief description of the backup. Enclose this description in single quotation marks (' '). This identifier is appended to the named keystore file (for example, ewallet_time_stamp_emp_key_backup.p12, with emp_key_backup being the backup identifier).

8. Skip this step, if wallet is created on shared location. Copy the ewallet.p12 file and backup file created in the directory "`/etc/oracle/wallet/ORCL1RAC`" from RAC1 to RAC2 and RAC3 in the same directory as on RAC1.

```
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet*
oracle@rac2.localdomain:/etc/oracle/wallet/ORCL1RAC/
```

```
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet*
oracle@rac3.localdomain:/etc/oracle/wallet/ORCL1RAC/
```

To verify the master encryption key is encrypting the RAC configuration

1. Create a CUSTOMERS table in the database.

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

2. Enter some values in the CUSTOMERS table.

```
SQL> INSERT INTO CUSTOMERS VALUES (001, 'George Bailey', 10000);
```

```
SQL> INSERT INTO CUSTOMERS VALUES (002, 'Denial Vettory', 20000);
```

```
SQL> INSERT INTO CUSTOMERS VALUES (003, 'MS Dhoni', 30000);
```

```
SQL> INSERT INTO CUSTOMERS VALUES (004, 'Shahid Afridi', 40000);
```

3. Encrypt the 'CREDIT_LIMIT' column of the 'CUSTOMERS' table.

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

4. With the next command, the values listed in the encrypted column are returned in clear text; Transparent Data Encryption decrypts them automatically.

```
SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;
```

5. List encrypted columns in your database.

```
SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;
```

6. View information about the software keystore.

```
SQL> SELECT * FROM GV$ENCRYPTION_WALLET;
```

7. Create an encrypted tablespace.

```
SQL> CREATE TABLESPACE SECURESPACE DATAFILE '+DATA' SIZE 150M ENCRYPTION
DEFAULT STORAGE (ENCRYPT);
```

8. Create a table in the tablespace.

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

9. Insert some values in EMPLOYEE table.

```
SQL> INSERT INTO EMPLOYEE VALUES (001,'JOHN SMITH',15000);

SQL> INSERT INTO EMPLOYEE VALUES (002,'SCOTT TIGER',25000);

SQL> INSERT INTO EMPLOYEE VALUES (003,'DIANA HAYDEN',35000);
```

10. Display the contents of the EMPLOYEE table with the following command.

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

11. Close the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<software_keystore_password>;
```

12. After closing the keystore, execute the following command to display the contents again:

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

If the keystore is closed, you will get the following error indicating that you cannot list the contents of EMPLOYEE table.

```
ERROR at line 1:

ORA-28365: wallet is not open
```

13. Now connect to RAC2 and RAC3 machine and execute the commands below after opening the wallet.

    i. Open the keystore.

    ```
    SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
    <software_keystore_password>;
    ```

    ii. Access the contents of EMPLOYEE table.

```
SQL> SELECT * FROM EMPLOYEE;

SQL> exit
```

Each time when you start the database, you need to open the wallet to view the encrypted data. You can set the wallet to Auto-Login so that it automatically opens when database starts.

> **NOTE:** The above commands works on all three nodes —RAC1, RAC2 and RAC3— after copying the wallet on all instances.

Test if the database can reach the HSM

1. Refer to Appendix B and complete the Keystore for Migration from Software to Hardware configuration based on the installed Oracle Database version.

> Configure Keystore for Migration from Software to Hardware for Oracle Database 12c

> Configure Keystore for Migration from Software to Hardware for Oracle Database 18c and 19c

2. Connect to sqlplus.

```
$ sqlplus / as sysdba
```

3. Connect to the database as 'system'.

```
SQL> connect system/<password>
```

4. Migrate the wallet onto the HSM.

```
SQL> ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY
"hsm_partition_pwd" MIGRATE USING <software_keystore_password> WITH BACKUP
USING 'backup_identifier';
```

> **NOTE:** "hsm_partition_pwd" is the password for the HSM partition. The 'MIGRATE USING software_keystore' string re-encrypts the Transparent Data Encryption column keys and tablespace keys with the new HSM based master key. The <software_keystore_password> is the password of software wallet.

5. If not using the shared location, Copy the ewallet.p12 and backup files created in the directory "`/etc/oracle/wallet/ORCL1RAC`" from RAC1 to RAC2 and RAC3 in the same directory as RAC1.

```
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet*
oracle@rac2.localdomain:/etc/oracle/wallet/ORCL1RAC/

$ scp /etc/oracle/wallet/ORCL1RAC/ewallet*
oracle@rac3.localdomain:/etc/oracle/wallet/ORCL1RAC/
```

6. Change the password of software keystore to same as HSM partition password.

```
SQL> ADMINISTER KEY MANAGEMENT ALTER KEYSTORE PASSWORD IDENTIFIED BY
<software_keystore_password> SET hsm_partition_pwd WITH BACKUP USING
'backup_identifier';
```

7. Copy the ewallet.p12 and backup files created in the directory "`/etc/oracle/wallet/ORCL1RAC`" from RAC1 to RAC2 and RAC3 in the same directory as RAC1.

   From now onwards when you open the keystore, it will open both the software-based keystore as well as the HSM-based keystore.

8. Restart the database.

```
$ srvctl stop database -d ORCL1RAC
$ srvctl start database -d ORCL1RAC
```

9. Connect to sqlplus.

```
$ sqlplus / as sysdba
```

10. Connect to the database as 'system'.

```
SQL> connect system/<password>
```

11. Open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
"hsm_partition_pwd";
```

12. With the next command, the values listed in the encrypted column are returned in clear text; Transparent Data Encryption decrypts them automatically, now using the HSM master key.

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

13. Check the wallet information with the following command.

```
SQL> SELECT * FROM GV$ENCRYPTION_WALLET;
```

> **NOTE:** After copying the wallet on all RAC instances, the preceding commands will work on all three nodes. You must copy the wallet from RAC1 instance to others after changing the wallet password. This is not needed if you are using the shared location for wallet.

14. Create the auto-login wallet on all the instances. Change the password back to the initial password for software based wallet (if you wish to change but it is preferred to keep the same password for both wallets) and use the following syntax to create an auto-login keystore for a software keystore:

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
IDENTIFIED BY <software_keystore_password>;
```

> **NOTE:** For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE 'keystore_location' IDENTIFIED BY <software_keystore_password>;** where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.
>
> **NOTE:** To use the auto-login wallet only on local system include LOCAL AUTO_LOGIN instead of AUTO_LOGIN.

15. Verify that an auto-open software keystore has been created in the oracle wallet directory you specified in the `sqlnet.ora` file. You will find two wallets in this directory: "ewallet.p12" and "cwallet.sso"; the latter is the auto-open wallet. Rename the ewallet.p12 to ewallet.p24 so that Oracle picks only cwallet.sso when auto login is enabled.

> **NOTE:** Copy the auto-login wallet and encryption wallet from one RAC instance to others after creating the wallet. It is not required if you use shared storage for the wallet.
>
> Renaming the wallet is required for v12.2.0.1 only but it is recommended to rename the encryption wallet when auto login wallet has been created.

16. Restart the database.

```
$ srvctl stop database -d ORCL1RAC
$ srvctl start database -d ORCL1RAC
```

17. Connect to the database as system and open the HSM keystore. The software wallet will open automatically.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
"hsm_partition_pwd";
```

To create the HSM Auto Wallet when HSM and Auto-Open Software wallet is in use

When the database restarts, the software wallet opens automatically, but the HSM based wallet needs to be opened manually.

1. Refer to Appendix B and complete the Software keystore configuration based on the installed Oracle Database version:

> Configure Software Keystore for Oracle Database 12c

> Configure Software Keystore for Oracle Database 18c and 19c

2. Rename the cwallet.sso on all nodes.
3. Ensure that the encryption wallet ewallet.p12 is present on all nodes.
4. Restart the database, connect as a system, and then open the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<software_keystore_password>;
```

5. Add the HSM secret as a client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'hsm_partition_password' FOR
CLIENT 'HSM_PASSWORD' IDENTIFIED BY <software_keystore_password> WITH
BACKUP USING 'backup_identifier';
```

   The secret is the hardware security module password and the client is the HSM_PASSWORD. HSM_PASSWORD is an Oracle-defined client name that is used to represent the HSM password as a secret in the software keystore.

6. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<software_keystore_password>;
```

7. Create (or recreate) auto-login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
IDENTIFIED BY <software_keystore_password>;
```

   NOTE: For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE 'keystore_location' IDENTIFIED BY <software_keystore_password>;** where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.

8. Rename the ewallet.p12 to ewallet.p24 so that Oracle picks only cwallet.sso when auto login is enabled.

> **NOTE:** Copy the auto-login wallet and encryption wallet from one RAC instance to others after creating the wallet. It is not required if you use shared storage for the wallet.
>
> Renaming the wallet is required for v12.2.0.1 only but it is recommended to rename the encryption wallet when auto-login wallet has been created.

9. Refer to Appendix B and complete the Keystore for Migration from Software to Hardware configuration based on the installed Oracle Database version:

> Configure Keystore for Migration from Software to Hardware for Oracle Database 12c

> Configure Keystore for Migration from Software to Hardware for Oracle Database 18c and 19c

10. Restart the database and connect as a system.
11. Check the wallet information with the following command:

```
SQL> SELECT * FROM GV$ENCRYPTION_WALLET;
```

Generating the Master Encryption Key directly on the HSM

> **NOTE:** It is assumed that no software or HSM based wallet is yet created.

1. Refer to Appendix B and complete the hardware keystore configuration based on the installed Oracle Database version.

> Configure Hardware Keystore for Oracle Database 12c

> Configure Hardware Keystore for Oracle Database 18c and 19c

2. If the database is not yet started, restart the database.

```
$ srvctl stop database -d ORCL2RAC
```

```
$ srvctl start database -d ORCL2RAC
```

Check the status when database start command completes.

```
$ srvctl status database -d ORCL2RAC
```

```
--------------------------------------------------------------------------------
```

```
Instance orcl2rac1 is running on node rac1
```

```
Instance orcl2rac2 is running on node rac2
```

```
Instance orcl2rac3 is running on node rac3
```

```
--------------------------------------------------------------------------------
```

3. Start the database.

```
$ sqlplus / as sysdba
```

4. Grant ADMINISTER KEY MANAGEMENT or SYSKM privilege to SYSTEM and any user that you want to use.

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

```
SQL> commit;
```

5. Connect to the database as 'system'.

```
SQL> connect system/<password>
```

> **NOTE:** Password for 'system' can be set during Oracle installation. All dbapasswords throughout this document has been set to "Temp1234".

6. Run the ADMINISTER KEY MANAGEMENT SQL statement to open the HSM keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
"hsm_partition_password";
```

7. Set the master encryption key in the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY
"hsm_partition_password";
```

You can see the HSM partition contents to verify the generated keys on HSM.

See below for an example snapshot of HSM partition contents:

```
Partition Name:  ORCL2
      Partition SN:    152042028
      Storage (Bytes): Total=102701, Used=1848, Free=100853
      Number objects:  5
      Object Label:  ORACLE.TDE.HSM.MK.0661286A8C71864F2ABF7891D044154D9A
      Object Type:   Symmetric Key
      Object Label:  DATA_OBJECT_SUPPORTED_IDEN
      Object Type:   Data
      Object Label:
ORACLE.SECURITY.KM.ENCRYPTION.303636313238364138433731383634463241424637383931443034343134353443941
      Object Type:   Data
      Object Label:  DATA_OBJECT_SUPPORTED_IDEN
      Object Type:   Data
      Object Label:
ORACLE.TSE.HSM.MK.072AC159D9153C4FF0BF3BF931ED9693850203
      Object Type:   Symmetric Key
```

To verify the master encryption key is encrypting the RAC configuration

1. Create a CUSTOMERS table in the database.

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

2. Enter some values in the CUSTOMERS table.

```
SQL> INSERT INTO CUSTOMERS VALUES (001, 'George Bailey', 10000);
```

```
SQL> INSERT INTO CUSTOMERS VALUES (002, 'Denial Vettory', 20000);

SQL> INSERT INTO CUSTOMERS VALUES (003, 'MS Dhoni', 30000);

SQL> INSERT INTO CUSTOMERS VALUES (004, 'Shahid Afridi', 40000);
```

3. Encrypt the 'CREDIT_LIMIT' column of the 'CUSTOMERS' table.

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

4. With the next command, the values listed in the encrypted column are returned in clear text. Transparent Data Encryption decrypts them automatically.

```
SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;
```

5. List encrypted columns in your database.

```
SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;
```

6. View information about the software keystore.

```
SQL> SELECT * FROM GV$ENCRYPTION_WALLET;
```

7. Create an encrypted tablespace.

```
SQL> CREATE TABLESPACE SECURESPACE DATAFILE '+DATA' SIZE 150M ENCRYPTION
DEFAULT STORAGE (ENCRYPT);
```

8. Create a table in the tablespace.

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

9. Insert some values in EMPLOYEE table.

```
SQL> INSERT INTO EMPLOYEE VALUES (001,'JOHN SMITH',15000);

SQL> INSERT INTO EMPLOYEE VALUES (002,'SCOTT TIGER',25000);

SQL> INSERT INTO EMPLOYEE VALUES (003,'DIANA HAYDEN',35000);
```

10. Display the contents of the EMPLOYEE table with the following command.

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

11. Close the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
"hsm_partition_password";
```

12. Display the contents again.

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

    If the keystore is closed, you will get the following error indicating that you cannot list the contents of EMPLOYEE table.

```
ERROR at line 1:

ORA-28365: wallet is not open
```

13. Open the wallet on the RAC2 and RAC3 machines.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
"hsm_partition_password";
```

14. Try to access the contents of EMPLOYEE table.

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

Now when you start the database, you need to open the HSM based wallet to view the encrypted data. You can set the HSM based wallet for auto login so that it will automatically open when the database starts.

To create the HSM auto wallet

1. Close the hardware security module if it is open.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
"hsm_partition_password";
```

2. Refer to Appendix B and complete the Software keystore configuration based on the installed Oracle Database version:

> Configure Software Keystore for Oracle Database 12c

> Configure Software Keystore for Oracle Database 18c and 19c

3. Create the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY
<software_keystore_password>;
```

> **NOTE:** For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'keystore_location' IDENTIFIED BY <software_keystore_password>;** where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.

> **NOTE:** If a software wallet has already been created, you need to skip this step and remove/rename the cwallet sso file.

4. Open the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<software_keystore_password>;
```

5. Add the HSM secret as a client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'hsm_partition_password' FOR
CLIENT 'HSM_PASSWORD' IDENTIFIED BY <software_keystore_password> WITH
BACKUP USING 'backup_identifier';
```

6. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<software_keystore_password>;
```

7. Create Auto-Login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
IDENTIFIED BY <software_keystore_password>;
```

> **NOTE:** For Oracle Database 12c use **ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE 'keystore_location' IDENTIFIED BY**

<software_keystore_password>; where 'keystore_location' is the path to the oracle wallet directory that you set in the sqlnet.ora file.

8. Optionally, rename the ewallet.p12 to ewallet.p24 so that Oracle picks only cwallet.sso when auto login is enabled.
9. Refer to Appendix B and complete the Keystore for Migration from Software to Hardware configuration based on the installed Oracle Database version:

> Configure Keystore for Migration from Software to Hardware for Oracle Database 12c

> Configure Keystore for Migration from Software to Hardware for Oracle Database 18c and 19c

10. If you are not using shared storage to create the wallet, copy both ewallet.p12 and cwallet.sso files created in the directory "`/etc/oracle/wallet/ORCL2RAC`" from RAC1 to RAC2 and RAC3 in the same directory as on RAC1.

```
$ scp /etc/oracle/wallet/ORCL2RAC/*
oracle@rac2.localdomain:/etc/oracle/wallet/ORCL2RAC/
```

```
$ scp /etc/oracle/wallet/ORCL2RAC/*
oracle@rac3.localdomain:/etc/oracle/wallet/ORCL2RAC/
```

    At this stage, close the wallet and open it one more time. The next time a TDE operation executes, the hardware security module auto-login keystore opens automatically.

11. Restart the database and connect as a system.
12. Check the wallet information with the following command:

```
SQL> SELECT * FROM GV$ENCRYPTION_WALLET;
```

Working with Pluggable Databases (PDB)

Refer to the “Working with Pluggable Databases (PDB)” section of Chapter 2 for detailed steps.

Ensure that the changes corresponding to **tnsnames.ora** file are made on all RAC instances (RAC1, RAC2 and RAC3).

CHAPTER 4: Integrating Luna HSM with Oracle Data Guard Physical Standby

This chapter covers the following topics:

Oracle Physical Standby Database with Transparent Data Encryption

Oracle Physical Standby encrypts redo logs while transferring between the primary and standby databases.

The master key from the primary database must be present on the Standby Site when the Standby Site is in READ ONLY mode or after a failover, but not for applying the redo logs. We recommend copying the primary wallet to the Standby sites, so that in the event of a failover all data will remain available.

Using HSM Wallet with Standby Oracle database

This section describes the steps that you need to follow when integrating a Standby Database with a Primary database where TDE is already enabled. It addresses the following two scenarios:

- Primary database software wallet used for TDE and existing master key migrated to HSM
- Primary database master key created directly on HSM

Prerequisites

Before you get started, verify the following details:

- Luna Client has been installed on both Primary database and Standby database.
- NTLS connection exists on both Primary and Standby database.
- Primary and Standby database are accessing the same HSM partition.
- Luna HSM PKCS#11 library is copied to the directory structure recommended by Oracle to ensure that the database is able to find this library.

  Use the following directory structures for UNIX and Windows, respectively:

Linux/Solaris/ AIX	`"/opt/oracle/extapi/[32,64]/hsm/{Vendor}/{Version}/libapiname.ext"`

Windows	"%SYSTEMDRIVE%\oracle\extapi\[32,64]\hsm\{Vendor}\{Version}\libapiname.ext"

For example: /opt/oracle/extapi/64/hsm/safenet/<7.x.x>/libCryptoki2_64.so

Oracle Database 12c

Scenario 1: Master key migrated from software sallet to HSM wallet

Steps to follow on Primary Database

This procedural set presumes that the software wallet is already in use, and that the software wallet was used to create the encrypted tables and tablespaces. To create TDE column and tablespace encryption using software wallet, refer to the section "Verify that the 'traditional' software-based wallet is working fine".

To verify the primary database encryption is operational

1. Verify that the sqlnet.ora file appears as follows:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA =
(DIRECTORY = <path to the oracle wallet directory>)))
```

2. Create and encrypt a test tablespace.

```
SQL> CREATE TABLESPACE securespace DATAFILE
'/home/oracle/app/oracle/oradata/orcl/secure01.dbf' SIZE 10M ENCRYPTION
DEFAULT STORAGE (ENCRYPT);
```

3. Create a table in the tablespace.

```
SQL>create table employee (id number(5), name varchar(42), salary
number(10)) TABLESPACE securespace;
```

4. Insert some values in the employee table.

```
SQL> Insert into employee values (001, 'JOHN SMITH', 10000);

SQL> Insert into employee values (002, 'SCOTT TIGER', 20000);

SQL> Insert into employee values (003, 'DIANA HAYDEN', 50000);
```

To migrate the master encryption key from the software wallet to the HSM

1. Change sqlnet.ora as below:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA =
(DIRECTORY = <path to the oracle wallet directory>)))
```

2. Connect to the database as 'system'.
3. Migrate the wallet onto the HSM device.

```
SQL> ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY
"hsm_partition_pwd" MIGRATE USING <software_keystore_password> WITH BACKUP
USING 'backup_identifier';
```

> **NOTE:** "hsm_partition_pwd" is the password for the HSM partition where the Master Encryption Key would be generated. The “migrate using <software_keystore_password>” string re-encrypts the Transparent Data Encryption column keys and tablespace keys with the new HSM based master key.

4. Now, test if you are able to execute queries on encrypted tablespace, for example,

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

The command returns employee information.

Create the HSM Auto-open wallet using the below steps. If you do not wish to create auto-open wallet, proceed to the section “Steps to follow on Standby Database”.

a. Change the sqlnet.ora entries as follows:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA =
(DIRECTORY = /etc/oracle/wallet)))
```

> **NOTE: “**/etc/oracle/wallet” is the directory where the encryption wallet has been generated.

b. Connect to the database as system and open the software keystore.

```
SQL> connect system/<password>
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<software_keystore_password>;
```

> **NOTE:** If an error is returned, “ORA-28354: Encryption wallet, auto login wallet, or HSM is already open”, restart the database and connect as system. Re-attempt executing the command.

c. Add the HSM secret as a client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'hsm_partition_password' FOR
CLIENT 'HSM_PASSWORD' IDENTIFIED BY <software_keystore_password> WITH
BACKUP USING 'backup_identifier';
```

> **NOTE:** The secret is the hardware security module password and the client is the HSM_PASSWORD.
>
> HSM_PASSWORD is an Oracle-defined client name that is used to represent the HSM password as a secret in the software keystore.

d. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<software_keystore_password>;
```

> **NOTE:** If an error “ORA-28374: typed master key not found in wallet” is returned, ignore it and execute next step to create Auto-Login keystore.

e. Create (or recreate) Auto-Login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'keystore_location' IDENTIFIED BY <software_keystore_password>;
```

f. Update the sqlnet.ora file to use the hardware security module.

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA =
(DIRECTORY = /etc/oracle/wallet)))
```

g. Restart the database and connect as a system.

h. Check the wallet information with the following command

```
SQL> SELECT * FROM V$ENCRYPTION_WALLET;
```

The HSM wallet will open automatically when database restarts. No password is required to access encrypted data with TDE master encryption key stored in the HSM.

Steps to follow on Standby Database

It is assumed that Standby database is already configured successfully as per the instructions provided in the "Prerequisites" section.

To configure the Standby database to integrate with the Primary

1. Create the same wallet directory as the one created in the Primary Database and copy all contents of the wallet directory from the Primary database to the wallet directory on the Standby database.

```
# scp -r oracle@<hostname or IP of Primary>:/etc/oracle/wallet/*
/etc/oracle/wallet
```

> **NOTE:** Assuming "/etc/oracle/wallet" is the directory where the encryption wallet has been generated.

2. Copy sqlnet.ora file from Primary Database to Standby Database at $ORACLE_HOME/network/admin/ location:

```
scp oracle@<hostname or IP of
Primary>:$ORACLE_HOME/network/admin/sqlnet.ora
$ORACLE_HOME/network/admin/sqlnet.ora
```

Its content should be:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA =
(DIRECTORY = <Path to oracle wallet>)))
```

To configure the Standby database to support the Primary

1. Start the database.

```
$ sqlplus / as sysdba
```

2. Resume managed recovery.

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM
SESSION;
```

All information about the HSM wallet is now provided to the Standby database. When standby database recovery is completed, it can be opened in read-only mode to allow query access.

3. Switch the standby database into read-only mode.

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE OPEN READ ONLY;
```

4. Connect the database as system and execute the query on your encrypted tablespace. For example, display the contents of the EMPLOYEE table with the following command:

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

It will return

```
ID NAME SALARY
---------- ---------------------------------------- ----------
1 JOHN SMITH 10000
2 SCOTT TIGER 20000
3 DIANA HAYDEN 50000
```

NOTE: If the HSM Auto Wallet is not configured on Primary database after migrating the key from software wallet to the HSM, open the wallet before running select command on encrypted table or tablespaces on standby.

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "hsm_partition_pwd" ;

Scenario 2: Master key was generated directly on HSM

To verify the primary database encryption is operational

1. Verify the sqlnet.ora file appears as follows:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))
```

2. Create and encrypt a test tablespace.

```
SQL> CREATE TABLESPACE securespace DATAFILE
'/home/oracle/app/oracle/oradata/orcl/secure01.dbf' SIZE 10M ENCRYPTION
DEFAULT STORAGE (ENCRYPT);
```

3. Create a table in the tablespace.

```
SQL>create table employee (id number(5), name varchar(42), salary
number(10)) TABLESPACE securespace;
```

4. Insert some values in the employee table.

```
SQL> Insert into employee values (001, 'JOHN SMITH', 10000);

SQL> Insert into employee values (002, 'SCOTT TIGER', 20000);

SQL> Insert into employee values (003, 'DIANA HAYDEN', 50000);
```

5. Verify the wallet status.

```
SQL> select * from v$encryption_wallet;
```

6. If you do not wish to use HSM Auto Wallet, copy the sqlnet.ora file from Primary Database to Standby Database at the following location on standby:

   `$ORACLE_HOME/network/admin/sqlnet.ora`

   and then proceed to the section, "Steps to follow on Standby Database"

To create the HSM Auto wallet on Primary database

1. Close the hardware security module, if it is open.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
"hsm_partition_password";
```

2. Change the sqlnet.ora entries as follows:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA =
(DIRECTORY = <Path to oracle wallet>)))
```

   For example:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA =
(DIRECTORY = /etc/oracle/wallet)))
```

3. Create the software keystore in the appropriate location (for example, "`/etc/oracle/wallet`").

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/oracle/wallet'
IDENTIFIED BY <software_keystore_password>;
```

4. Open the software keystore,

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<software_keystore_password>;
```

5. Add the HSM password as a client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'hsm_partition_password' FOR
CLIENT 'HSM_PASSWORD' IDENTIFIED BY <software_keystore_password> WITH
BACKUP USING 'backup_identifier';
```

> **NOTE:** The secret is the hardware security module password and the client is the HSM_PASSWORD.
> HSM_PASSWORD is an Oracle-defined client name that is used to represent the HSM password as a secret in the software keystore.

6. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
<software_keystore_password>;
```

> **NOTE:** If the error ORA-28374: typed master key not found in wallet" is returned, ignore it and execute next step to create Auto-Login keystore.

7. Create Auto-Login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'/etc/oracle/wallet' IDENTIFIED BY <software_keystore_password>;
```

8. Update the `sqlnet.ora` file to use the hardware security module.

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA =
(DIRECTORY = /etc/oracle/wallet)))
```

9. Restart the database and connect as a system.
10. Check the wallet information with the following command:

```
SQL> SELECT * FROM V$ENCRYPTION_WALLET;
```

The HSM wallet will now open automatically when the database restarts. No password is required to access encrypted data with the TDE master encryption key stored in the HSM.

To manage database recovery on the Standby database

1. Ensure you follow the prerequisites listed in the section "Prerequisites".
2. If you created the auto wallet on the primary database, create the same primary database wallet directory and copy all contents of the wallet directory from the primary database to the wallet directory on the standby database.
3. Copy the sqlnet.ora file from the primary database to the standby database at the following location:

   $ORACLE_HOME/network/admin/sqlnet.ora

   Its content should be:

   For HSM Only Wallet (No Auto Login)

   ```
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))
   ```

   For HSM with Auto Wallet

   ```
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA =
   (DIRECTORY = <Path to oracle wallet>)))
   ```

4. Connect to the standby database as sysdba.

   ```
   $ sqlplus / as sysdba
   ```

5. Resume managed recovery.

   ```
   SQL> SHUTDOWN IMMEDIATE;
   SQL> STARTUP MOUNT;
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM
   SESSION;
   ```

   When the standby database recovery is complete, the standby database can be opened in read-only mode to allow query access.

6. Switch the standby database to read-only mode.

   ```
   SQL> SHUTDOWN IMMEDIATE;
   SQL> STARTUP MOUNT;
   SQL> ALTER DATABASE OPEN READ ONLY;
   ```

7. Execute a test query on the Standby database to verify it can access the encrypted tablespace.

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

   It will return:

   ```
   ID NAME SALARY
   ---------- ---------------------------------------- ----------
   ```

```
1 JOHN SMITH 10000
2 SCOTT TIGER 20000
3 DIANA HAYDEN 50000
```

NOTE: If the HSM Auto Wallet is not configured on primary, open the wallet before running select command on encrypted table or tablespaces in standby.

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "hsm_partition_pwd" ;

APPENDIX A: Troubleshooting Tips

Problem 1

While generating the master key on the HSM or migrate keys from the wallet to the HSM, you see the following error message:

ORA-43000: PKCS11: library not found

Or,

ORA-28376: cannot find PKCS11 library

Solution

1. Ensure that the library path is set correctly:

   ```
   "/opt/oracle/extapi/[32/64]/HSM/[x.x.x]/libcryptoki2_64.so"
   ```

2. Ensure that oracle : oinstall is in the owner : group of the above directory with read/write permissions.
3. Ensure that the 64-bit JVM is running on the machine on which we are using 64-bit client because 32-bit JVM would not able to use the 64-bit library.

Problem 2

When opening the keystore or generating the Master Key on the HSM, you encounter the following error message in PDB database:

ORA-46627: keystore password mismatch

Solution

1. Ensure that the provided HSM password is correct.
2. Ensure that HSM Auto_Login or Local_Auto_Login is not enabled for the CDB.
3. If Auto_Login or Local_Auto_Login is enabled, then use the encryption wallet instead of auto wallet in CDB then perform this operation in PDB. After generating the Master Key for PDB you can enable the Auto_Login again in CDB and it works for all PDBs as well.

Problem 3

Wallet closes when using Oracle (12.1.0.1 / 12.1.0.2) on Windows and Solaris and oracle alert.log displays the following error message:
ORA-28407: Hardware Security Module failed with PKCS#11 error CKR_CRYPTOKI_ALREADY_INITIALIZED(%d)

Solution

Apply the related Solaris or Windows patch.

Windows – DOW4648

Solaris - DOW0002488

Problem 4

Wallet closes when using the Oracle 12.1.0.2 in DPoD and oracle alert.log is showing:
kzthsmcc encountered: ORA-28407: Hardware Security Module failed with PKCS#11 error CKR_TOKEN_NOT_PRESENT(224)
kzthsmcc1: HSM heartbeat check failed to cache
object handle. Error code: 1
HSM connection lost, closing wallet
2018-08-31T18:06:16.493157+05:30

Solution

Change the Oracle setting to wait for 30 seconds before closing the wallet. There is an event (Event 28420) which can be set to control "how many HSM heartbeats can fail before the wallet is closed."

To change the event settings, perform the following command:

```
ALTER SYSTEM SET EVENT='28420 trace name context forever, level 10' COMMENT='HSM
heartbeat timeout attempt' SCOPE=SPFILE;
```

> **NOTE:** You can change the trace level from 10 to any other value of your choice.

APPENDIX B: Setting Keystore on Oracle Database

Setting keystore on Oracle Database 12c

To set software keystore

Add the following to your **"$ORACLE_HOME/network/admin/sqlnet.ora"** file:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA =
(DIRECTORY = <path to the oracle wallet directory>)))
```

To set hardware keystore

Add the following to your **"$ORACLE_HOME/network/admin/sqlnet.ora"** – file:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))
```

To migrate from software to hardware

Add the following to your **"$ORACLE_HOME/network/admin/sqlnet.ora"** file:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA =(DIRECTORY
= <path to the oracle wallet directory>)))
```

Setting keystore on Oracle Database 18c and 19c

NOTE: Beginning with 19c release, Oracle recommends that you use the WALLET_ROOT and TDE_CONFIGURATION parameters to configure the keystore location, but you also have the option of configuring the sqlnet.ora file.

Oracle recommends using the initialization parameters in multitenant environment. Only united mode is supported if sqlnet.ora is used for TDE configuration in the multitenant environment. Isolated mode is supported only if the WALLET_ROOT and TDE_CONFIGURATION parameters are used for TDE configuration.

To set software keystore

1. Create a wallet directory located in the **$ORACLE_BASE/admin/db_unique_name** directory or any other location and it is named as **wallet**.
2. Log in to the database instance as a user who has been granted the `SYSDBA` administrative privilege.

   ```
   Connect / as sysdba
   ```

3. Set **WALLET_ROOT** parameter.

```
alter system set wallet_root='<path to the oracle wallet directory>'
scope=spfile;
```

4. Shut down and start up database.

```
shutdown immediate;
startup;
```

5. Set **TDE_CONFIGURATION** parameter.

```
alter system set TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=FILE" SCOPE=both;
```

To set hardware keystore

1. Create a wallet directory located in the **$ORACLE_BASE/admin/db_unique_name** directory or any other location and it is named as **wallet**.
2. Log in to the database instance as a user who has been granted the `SYSDBA` administrative privilege.

```
Connect / as sysdba
```

3. Set **WALLET_ROOT** parameter.

```
alter system set wallet_root='<path to the oracle wallet directory>'
scope=spfile;
```

4. Shutdown and startup database.

```
shutdown immediate;
startup;
```

5. Set **TDE_CONFIGURATION** parameter.

```
alter system set TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=HSM" SCOPE=both;
```

To migrate from software to hardware

1. Create a wallet directory located in the **$ORACLE_BASE/admin/db_unique_name** directory or any other location and it is named as **wallet**.
2. Log in to the database instance as a user who has been granted the `SYSDBA` administrative privilege.

```
Connect / as sysdba
```

3. Set **WALLET_ROOT** parameter.

```
alter system set wallet_root='<path to the oracle wallet directory>'
scope=spfile;
```

4. Shut down and start up database.

```
shutdown immediate;
startup;
```

5. Set **TDE_CONFIGURATION** parameter.

```
alter system set TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=HSM|FILE"
SCOPE=both;
```

APPENDIX C: Known Issues

Oracle Database Version	Platform	Oracle Known Issues	Oracle Patch/Workaround
19c (19.3.0.0)	Generic	PDB Auto-login HSM configurations Fail to Open the TDE Keystore	Patch 29834717: DATABASE RELEASE UPDATE 19.4.0.0.0
19c (19.3.0.0)	Generic	PDB Auto Login Failed to open wallet when HSM is used.	Patch 29530515: AUTO-LOGIN HSM CONFIGURATIONS FAIL TO OPEN THE TDE KEYSTORE
18c (18.3.0.0)	Generic	PDB Auto Login Failed to open wallet when HSM is used.	Patch 30872794: DATABASE RELEASE UPDATE 18.10.0.0.0
18c (18.3.0.0)	Generic	PDB Auto Login Failed to open wallet when HSM is used.	Patch 29530515: AUTO-LOGIN HSM CONFIGURATIONS FAIL TO OPEN THE TDE KEYSTORE
12cR1 (12.1.0.2) (12.1.0.1)	Generic	PDB Auto Login Failed to open wallet when HSM is used.	Patch 20842388: AUTO-LOGIN HSM SUPPORT FOR PDBS
12cR1 (12.1.0.2) (12.1.0.1)	Generic	Key Migration failed when Auto Login Wallet is in use.	Patch 22826718: Online rekey patch
12cR1 (12.1.0.2) (12.1.0.1)	Generic	Oracle creates Data Objects every time when wallet opened.	Patch 23514911: C_CREATEOBJECT CALLED REPEATEDLY

Contacting Customer Support

If you encounter a problem while installing, registering, or operating this product, refer to the documentation. If you cannot resolve the issue, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support.DIS@thalesgroup.com.