All information herein is either public information or is the property of and owned solely by Gemalto and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Gemalto’s information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

- The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- This document shall not be posted on any publicly accessible network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided “AS IS” without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemalto makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemalto reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemalto hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemalto be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms’ resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

© 2008-18 Gemalto. All rights reserved. Gemalto and the Gemalto logo are trademarks and service marks of Gemalto and/or its subsidiaries and are registered in certain countries. All other trademarks and service marks, whether registered or not in specific countries, are the property of their respective owners.

**Document Number:** 007-008670-001, Rev. BE  
**Release Date:** July 2018
## Contents

Preface ........................................................................................................................................... 5
   Scope .......................................................................................................................................... 5
   Document Conventions ........................................................................................................... 5
       Command Syntax and Typeface Conventions .................................................................. 6
       Support Contacts ................................................................................................................. 7

1 Introduction ............................................................................................................................... 8
   Overview .................................................................................................................................... 8
       Third Party Application Details ......................................................................................... 8
       Supported Platforms ........................................................................................................... 9
       Multiple Database Support for Single Partition ............................................................... 10
       Known issues ....................................................................................................................... 11
   Prerequisites ............................................................................................................................ 12
       Configuring SafeNet Luna Network HSM .......................................................................... 12
       Provision your HSM on Demand Service ........................................................................... 13
       Set Flags for SafeNet HSM on Window Server 2012R2/2016 .......................................... 14
       Setup SafeNet Luna HSM HA (High-Availability) ............................................................... 14
       Setup Oracle Database ......................................................................................................... 14

2 Integrating SafeNet HSM with Oracle Database 12c ............................................................... 15
   Setting up Luna Client for Transparent Data Encryption ....................................................... 15
   Generating a Master Encryption Key for HSM-Based Encryption ......................................... 16
       Migrating Master Encryption Key onto the HSM ............................................................... 16
   Working with Pluggable Databases (PDB) ............................................................................ 24
       About Containers in a CDB ................................................................................................. 24

3 Integrating SafeNet HSM with Oracle Database 12c RAC ....................................................... 28
   Understanding Oracle RAC ..................................................................................................... 28
       Oracle Database RAC Setup ............................................................................................... 28
       Supported Platforms ........................................................................................................... 29
       Verifying the Oracle RAC Installation .............................................................................. 29
   Setting up SafeNet Luna HSM for Transparent Data Encryption with Oracle RAC ............... 34
       Configuring the PKCS11 Provider on Oracle RAC Instances ........................................... 34
       Migrating Master Encryption Key from software wallet to HSM ....................................... 35

4 Integrating SafeNet HSM with Oracle Data Guard Physical Standby .................................... 45
   Oracle Physical Standby database with Transparent Data Encryption .................................... 45
       Using HSM Wallet with Standby Oracle database ............................................................ 45
       Prerequisites ....................................................................................................................... 45
       Oracle Database 12c ............................................................................................................ 46
Scenario 1: Master key Migrated from Software Wallet to HSM wallet ........................................46
Scenario 2: Master key was generated directly on HSM .................................................................49

5 Troubleshooting Tips ..................................................................................................................52
This document will guide security administrators through the steps for integrating an Oracle Database with a SafeNet Luna HSM or HSM on Demand service. It also covers the necessary information to install, configure and integrate Oracle Database Transparent Data Encryption (TDE) with SafeNet HSM.

Scope

This document outlines the steps to integrate an Oracle Database with a SafeNet Luna HSM or HSM on Demand service. The SafeNet Luna HSM or the HSM on Demand service is used to secure the master encryption key for Oracle TDE.

Document Conventions

This section provides information on the conventions used in this template.

Notes

Notes are used to alert you to important or helpful information. These elements use the following format:

NOTE: Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. These elements use the following format:

CAUTION: Exercise caution. Caution alerts contain important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. These elements use the following format:

WARNING: Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.
## Command Syntax and Typeface Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
</table>
| **bold**   | The bold attribute is used to indicate the following:  
  - Command-line commands and options (Type `dir /p`).  
  - Button names (Click **Save As**.).  
  - Check box and radio button names (Select the **Print Duplex** check box.).  
  - Window titles (On the **Protect Document** window, click **Yes**.).  
  - Field names (**User Name**: Enter the name of the user.)  
  - Menu names (On the **File** menu, click **Save**.) (Click **Menu > Go To > Folders**.)  
  - User input (In the **Date** box, type **April 1**.) |
| **italic** | The italic attribute is used for emphasis or to indicate a related document. (See the **Installation Guide** for more information.) |
| *Consolas* | Denotes syntax, prompts, and code examples. |
## Support Contacts

<table>
<thead>
<tr>
<th>Contact Method</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Address</strong></td>
<td>Gemalto</td>
</tr>
<tr>
<td></td>
<td>4690 Millennium Drive</td>
</tr>
<tr>
<td></td>
<td>Belcamp, Maryland 21017, USA</td>
</tr>
<tr>
<td><strong>Phone</strong></td>
<td></td>
</tr>
<tr>
<td>US</td>
<td>1-800-545-6608</td>
</tr>
<tr>
<td>International</td>
<td>1-410-931-7520</td>
</tr>
<tr>
<td><strong>Technical Support</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Customer Portal</strong></td>
<td><a href="https://supportportal.gemalto.com">https://supportportal.gemalto.com</a></td>
</tr>
<tr>
<td>Existing customers with</td>
<td></td>
</tr>
<tr>
<td>a Technical Support</td>
<td>manage incidents, get the latest software upgrades,</td>
</tr>
<tr>
<td>Customer Portal</td>
<td>and access the Gemalto Knowledge Base.</td>
</tr>
</tbody>
</table>
Introduction

Overview

This document is intended to guide security administrators through integrating an Oracle Database with a SafeNet Luna HSM or an HSM on Demand Service, and also covers the necessary information to install, configure and integrate Oracle Database Transparent Data Encryption (TDE) with the HSM or service.

SafeNet HSMs come as on premise hardware HSMs widely known as SafeNet Luna HSM 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 SafeNet HSM or HSM on Demand service to secure the EKM for the following reasons:

- It stores the master encryption keys used for transparent data encryption, and the master encryption key is never exposed in insecure memory.
- It provides more secure computational storage.
- The HSM is a more secure alternative to the Oracle wallet.

Third Party Application Details

- Oracle Database 12c R2/R1
Supported Platforms

Below is the list of the platforms tested with the following HSMs:

**The SafeNet Luna HSM:** SafeNet 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. SafeNet Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing.

The SafeNet Luna HSM on premise offerings include the SafeNet Luna Network HSM, SafeNet PCIe HSM, and SafeNet Luna USB HSMs. SafeNet Luna HSMs are also available for access as an offering from cloud service providers such as IBM cloud HSM and AWS cloud HSM classic.

<table>
<thead>
<tr>
<th>Oracle Database</th>
<th>Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Database 12C</td>
<td>Windows Server 2016</td>
</tr>
<tr>
<td></td>
<td>Windows Server 2012 R2</td>
</tr>
<tr>
<td></td>
<td>RHEL</td>
</tr>
<tr>
<td></td>
<td>Solaris Sparc</td>
</tr>
<tr>
<td></td>
<td>Solaris x86</td>
</tr>
<tr>
<td></td>
<td>AIX</td>
</tr>
<tr>
<td></td>
<td>Oracle Linux</td>
</tr>
</tbody>
</table>

**NOTE:** This integration is tested with Luna Clients in HA and FIPS Mode.

**NOTE:** If you use Oracle Database 11g R1/R2 then refer to the previous version of the Oracle Database Integration guide (OracleDatabase_SafeNetLunaHSM_IntegrationGuide_RevBC).
SafeNet 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 just the services you need.

<table>
<thead>
<tr>
<th>Oracle Database</th>
<th>Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Database 12C</td>
<td>Windows Server 2016</td>
</tr>
<tr>
<td></td>
<td>Windows Server 2012 R2</td>
</tr>
</tbody>
</table>

### Multiple Database Support for Single Partition

<table>
<thead>
<tr>
<th>Oracle Version</th>
<th>Partition Configuration</th>
<th>Key Lifecycle</th>
<th>Encryption wallet creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1.0.1</td>
<td>Requires a unique dedicated partition per Oracle database/cluster</td>
<td>New Install</td>
<td>TDE and TSE keys are created</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Key Migrate</td>
<td>New TDE key is created but the TSE key remains the same</td>
</tr>
<tr>
<td>12.1.0.2</td>
<td>Supports both unique dedicated partitions per Oracle database/cluster and also a single shared partition configuration for all Oracle databases/clusters</td>
<td>New Install</td>
<td>Only TDE key is created</td>
</tr>
<tr>
<td>12.2.0.1</td>
<td></td>
<td>Key Migrate</td>
<td>New TDE key is created</td>
</tr>
</tbody>
</table>
# Known issues

<table>
<thead>
<tr>
<th>Oracle Database Version</th>
<th>Platform</th>
<th>Oracle Known Issues</th>
<th>Oracle Patch/Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>12cR1 (12.1.0.2)</td>
<td>Generic</td>
<td>PDB Auto Login Failed to open wallet when HSM is used.</td>
<td>Patch 20842388: AUTO-LOGIN HSM SUPPORT FOR PDBS</td>
</tr>
<tr>
<td>12cR1 (12.1.0.1)</td>
<td>Generic</td>
<td>Key Migration failed when Auto Login Wallet is in use.</td>
<td>Patch 22826718: Online rekey patch</td>
</tr>
<tr>
<td>12cR1 (12.1.0.2)</td>
<td>Generic</td>
<td>Oracle creates Data Objects every time when wallet opened.</td>
<td>Patch 23514911: C_CREATEOBJECT CALLED REPEATEDLY</td>
</tr>
</tbody>
</table>
Prerequisites

Before starting the integration of Oracle Database with SafeNet Luna HSM or HSM on Demand Service ensure you have completed configuring the SafeNet Luna Network HSM or provisioning HSM on Demand Service as per the requirement.

Configuring SafeNet Luna Network HSM

Before you get started with SafeNet Luna HSM ensure the following:

1. Ensure the HSM is setup, initialized, provisioned and ready for deployment. Refer to the HSM product documentation for help.
2. Create a partition on the HSM that will be later used by Oracle Database.
3. If using a SafeNet Luna Network HSM, register a client for the system and assign the client to the partition to create an NTLS connection. Initialize Crypto Officer and Crypto User roles for the registered partition. Ensure that the partition is successfully registered and configured. The command to see the registered partition is:

   `<lunacm utility path>/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 SafeNet Network Luna HSM documentation for detailed steps for creating NTLS connection, initializing the partitions and various user roles.

Controlling User Access to the HSM

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 your client software while retaining your hsmusers group configuration.

Adding users to hsmusers group

To allow non-root users or applications access to the HSM, assign the users to the hsmusers group. The users you assign to the hsmusers group must exist on the client workstation.
- **Adding user to hsmusers group**
  
a. Ensure that you have `sudo` privileges on the client workstation.
  
b. Add a user to the hsmusers group.
    ```bash
    sudo gpasswd --add <username> hsmusers
    ```
    Where `<username>` is the name of the user you want to add to the hsmusers group.

- **Removing users from hsmusers group**
  
  To revoke a user's access to the HSM, you can remove them from the hsmusers group.
  
  - **Removing a user from hsmusers group**
    
a. Ensure that you have `sudo` privileges on the client workstation.
    
b. Remove a user from the hsmusers group.
      ```bash
      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.

---

**Provision your HSM on Demand Service**

This service provides your client machine with access to 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.

To use the HSM on Demand service you need to provision your application partition, starting 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)** - optional role that can use crypto objects while performing cryptographic operations.

  **NOTE:** Refer to the “SafeNet Data Protection on Demand Application Owner Quick Start Guide” for procedural information on configuring the HSM on Demand service and create a service client.

  The HSM on Demand service client package is a zip file that contains system information needed to connect your client machine to an existing HSM on Demand service.
Constraints on HSMoD 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 the 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.

Set Flags for SafeNet HSM on Window Server 2012R2/2016
If you are using a SafeNet HSM 7.x with Windows Server 2012R2/2016 open the **Crystoki.ini** file in a text editor and set the following two flags in the Misc section:

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

Setup SafeNet Luna HSM HA (High-Availability)
Please refer to the SafeNet 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 goes down by some reason all calls automatically routed to secondary till primary gets up again.

Setup Oracle Database
Oracle Database must be installed on the target machine to carry on with the integration process. For a detailed installation procedure of Oracle Database, refer to the Oracle Database Documentation.
2 – Integrating SafeNet HSM with Oracle Database 12c

Integrating SafeNet HSM with Oracle Database 12c

Setting up Luna Client for Transparent Data Encryption

The following procedures will guide the user through setting up the HSM for Transparent Data Encryption (TDE).

To setup the Luna Client for TDE

Copy the SafeNet 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"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>[32,64]</td>
<td>Specifies whether the supplied binary is 32-bits or 64-bits.</td>
</tr>
<tr>
<td>[vendor]</td>
<td>Specifies the name of the vendor supplying the library.</td>
</tr>
</tbody>
</table>
| [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.

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=$ORACLE_BASE/product/12.1.0/dbhome_1
export PATH=$PATH:$ORACLE_HOME/bin
export TNS_ADMIN=$ORACLE_HOME/network/admin

Oracle Database Integration Guide
Generating 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. The HSM can be used in the following ways to protect the Master Encryption Key:

- An existing Unified Master Encryption Key can be migrated onto the HSM.
- A Unified Master Encryption Key can be directly generated onto the HSM.
- HSM Auto-Login wallet use for TDE.

Migrating Master Encryption Key onto the HSM

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

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

To test TDE with SafeNet HSM, perform the following steps:

To verify that the software-based wallet is working

1. 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>)))
   ```

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 'keystore_location' IDENTIFIED BY software_keystore_password;
   ```
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. The next command lists encrypted columns in your database:
   SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;

6. Finally, this view contains information about the software 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 software_keystore_password;

12. 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 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. Change your "$ORACLE_HOME/network/admin/sqlnet.ora" file:
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA =(DIRECTORY = <path to the
   oracle wallet directory>)))

2. $ sqlplus / as sysdba
   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.

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 'keystore_location'
       IDENTIFIED BY software_keystore_password;

   **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: 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";

---

**Creating the HSM Auto Wallet when HSM and Auto-Open Software wallet are in use**

1. Change the sqlnet.ora entries as follows:
   `ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet)))`

2. Rename or move the Auto-Open wallet from the location mentioned in the sqlnet.ora file and move or rename the encryption wallet in to the wallet directory.
   # mv ewallet.p24 ewallet.p12

3. Restart the database and connect as system.

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';
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;

7. Create (or recreate) Auto-Login keystore.
   SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE 'keystore_location'
      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. Verify the wallet information:
    SQL> SELECT * FROM V$ENCRYPTION_WALLET;

### Generating the Master Encryption Key directly on the HSM

To generate a Master Encryption Key for HSM-Based Encryption, complete the following:

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

### To setup Oracle to create the master encryption key directly on the HSM

1. Add the following to your "$ORACLE_HOME/network/admin/sqlnet.ora" – file:
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))

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 HSM keystore.
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "hsm_partition_password";
6. 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 an HSM by executing partition contents in lunacom.

---

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.30363631323836413836344632414246373839314430343431353
Object Type: Data
Object Label: DATA_OBJECT_SUPPORTED_IDEN
Object Type: Data
Object Label: ORACLE.TSE.HSM.MK.072AC159D9153C4FF0B3E931ED969350203
Object Type: Symmetric Key

---

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 software 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;
    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

13. Open the keystore:
    SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "hsm_partition_password";
    SQL> exit

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. Change the sqlnet.ora entries as follows:
   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';

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 '/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)))
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.

About Containers in a CDB

A container is either a PDB or the root container (also called the root). The root is a collection of schemas, schema objects, and non-schema objects to which all PDBs belong. Every CDB has the following containers:

Exactly one root: The root stores Oracle-supplied metadata and common users. A common user is a database user known in every container. The root container is named CDB$ROOT.

Exactly one seed PDB: The seed PDB is a system-supplied template that the CDB can use to create new PDBs. The seed PDB is named PDB$SEED. You cannot add or modify objects in PDB$SEED.

Zero or more user-created PDBs: A PDB is a user-created entity that contains the data and code required for a specific set of features. For example, a PDB can support a specific application, such as a human resources or sales application. No PDBs exist at creation of the CDB. You add PDBs based on your business requirements.

Purpose of PDBs

You can use PDBs to achieve the following goals:

1. Store data specific to a particular application. For example, a sales application can have its own dedicated PDB, and a human resources application can have its own dedicated PDB.

2. Move data into a different CDB. A database is "pluggable" because you can package it as a self-contained unit, and then move it into another CDB.

3. Isolate grants within PDBs. A local or common user with appropriate privileges can grant EXECUTE privileges on a package to PUBLIC within an individual PDB.

There are several ways to create a PDB but we recommend using the DBCA utility. It is assumed that you have already created PDBs. For the purpose of this guide we are using the PDB "salespdb".

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

2. Restart the Listener Service.
   
   ```
   # lsnrctl stop
   # lsnrctl start
   ```

3. Start the **sqlplus** session to connect to PDB.
   
   ```
   $ sqlplus / as sysdba
   ```
   
   SQL> alter pluggable database all open read write;
   Pluggable database altered.
   
   SQL> Connect system/<system_password>@Pluggable Database Service name
   For Example:
   
   SQL> connect system/temp123#@salespdb;

4. 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;
   Commit complete.
   ```
   Where, **salesadm** is the administrative user name created at the time of creating PDB.

5. 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
   Connected.
   ```

### To generate a TDE master encryption key for PDB

1. Add the following to your "$ORACLE_HOME/network/admin/sqlnet.ora" – file:
   
   ```
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))
   ```

2. 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
3. Run the ADMINISTER KEY MANAGEMENT SQL statement using the following syntax:
   
   ```sql
   ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "hsm_partition_password";
   ```

   For example:
   
   ```sql
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "userpin2";
   keystore altered.
   ```

   **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.

4. Create the PDB Master key:

   ```sql
   ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "hsm_partition_password";
   ```

   For example:
   
   ```sql
   SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "userpin2";
   keystore altered.
   ```

   **NOTE:** This will generate a new master key, any encryption/decryption operations performed within this PDB will use this master key.

---

**To verify the master encryption key is encrypting the PDB**

1. Create a CUSTOMERS table in the PDB.

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

2. Enter some values in the CUSTOMERS table.

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

5. The next command lists encrypted columns in your databases:

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

6. Create an encrypted tablespace:

   ```sql
   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
   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";
Understanding Oracle RAC

Oracle Real Application Clusters (Oracle RAC) enables an Oracle database to run across a cluster of servers, providing fault tolerance, performance, and scalability with no application changes necessary. Oracle RAC provides high availability for applications by removing the single point of failure with a single server.

Oracle Clusterware is installed into a single home directory, which is called the Grid home. Oracle Clusterware enables servers, referred to as hosts or nodes, to operate as if they are one server, commonly referred to as a cluster. Although the servers are standalone servers, each server has additional processes that communicate with other servers. In this way the separate servers appear as if they are one server to applications and end-users.

Non-cluster Oracle databases have a one-to-one relationship between the Oracle database and the instance. Alternatively, Oracle RAC environments, have a one-to-many relationship between the database and instances. Oracle RAC databases differ architecturally from non-cluster Oracle databases in that each Oracle RAC database instance also has:

- At least one additional thread of redo for each instance
- An instance-specific undo tablespace

The combined processing power of the multiple servers can provide greater throughput and Oracle RAC scalability than is available from a single server.

Oracle Database RAC Setup

We recommend you familiarize yourself with the Oracle Database RAC documentation. We recommend reading the *Oracle Database 12c RAC documentation* for more information about installing and pre-installation requirements.

The 3 machines utilized are denoted in the setup as follows:

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

This demonstration uses 3 Oracle Homes. Each Oracle Home has 1 database and 3 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 SafeNet Luna HSM:

**Oracle Database 12c**

<table>
<thead>
<tr>
<th>SafeNet HSM</th>
<th>Platforms Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>SafeNet Luna HSM</td>
<td>AIX</td>
</tr>
</tbody>
</table>

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
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Target</th>
<th>State</th>
<th>Server</th>
<th>State details</th>
</tr>
</thead>
<tbody>
<tr>
<td>ora.DATA.dg</td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac1</td>
<td>STABLE</td>
</tr>
<tr>
<td></td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac2</td>
<td>STABLE</td>
</tr>
<tr>
<td></td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac3</td>
<td>STABLE</td>
</tr>
<tr>
<td>ora.LISTENER.lsnr</td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac1</td>
<td>STABLE</td>
</tr>
<tr>
<td></td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac2</td>
<td>STABLE</td>
</tr>
<tr>
<td></td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac3</td>
<td>STABLE</td>
</tr>
<tr>
<td>ora.asm</td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac1</td>
<td>Started,STABLE</td>
</tr>
<tr>
<td></td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac2</td>
<td>Started,STABLE</td>
</tr>
<tr>
<td></td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac3</td>
<td>Started,STABLE</td>
</tr>
<tr>
<td>ora.net1.network</td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac1</td>
<td>STABLE</td>
</tr>
<tr>
<td></td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac2</td>
<td>STABLE</td>
</tr>
<tr>
<td></td>
<td>ONLINE</td>
<td>ONLINE</td>
<td>rac3</td>
<td>STABLE</td>
</tr>
<tr>
<td>Resource</td>
<td>Status</td>
<td>Node 1</td>
<td>Node 2</td>
<td>Node 3</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>ora.LISTENER_SCAN1.lsnr</td>
<td>ONLINE ONLINE</td>
<td>rac2</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.LISTENER_SCAN2.lsnr</td>
<td>ONLINE ONLINE</td>
<td>rac1</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.LISTENER_SCAN3.lsnr</td>
<td>ONLINE ONLINE</td>
<td>rac3</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.MGMTLSNR</td>
<td>ONLINE ONLINE</td>
<td>rac2</td>
<td>Open,STABLE</td>
<td></td>
</tr>
<tr>
<td>ora.cvu</td>
<td>ONLINE ONLINE</td>
<td>rac3</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.mgmtdb</td>
<td>ONLINE ONLINE</td>
<td>rac2</td>
<td>Open,STABLE</td>
<td></td>
</tr>
<tr>
<td>ora.oc4j</td>
<td>ONLINE ONLINE</td>
<td>rac3</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.orcl1rac.db</td>
<td>ONLINE ONLINE</td>
<td>rac1</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td></td>
<td>ONLINE ONLINE</td>
<td>rac2</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td></td>
<td>ONLINE ONLINE</td>
<td>rac3</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.orcl2rac.db</td>
<td>ONLINE ONLINE</td>
<td>rac1</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td></td>
<td>ONLINE ONLINE</td>
<td>rac2</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td></td>
<td>ONLINE ONLINE</td>
<td>rac3</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.orcl3rac.db</td>
<td>ONLINE ONLINE</td>
<td>rac1</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td></td>
<td>ONLINE ONLINE</td>
<td>rac2</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td></td>
<td>ONLINE ONLINE</td>
<td>rac3</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.rac1.vip</td>
<td>ONLINE ONLINE</td>
<td>rac1</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ora.rac2.vip</td>
<td>ONLINE ONLINE</td>
<td>rac2</td>
<td>ONLINE</td>
<td>ONLINE</td>
</tr>
</tbody>
</table>
The setup has 3 databases ORCL1RAC, ORCL2RAC, and ORCL3RAC and each database has 3 instances which run simultaneously on 3 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/pwдорcl3rac.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 SafeNet Luna HSM for Oracle TDE.

**Setting up SafeNet 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:

- An existing Unified Master Encryption Key can be migrated onto the HSM.
- A Unified Master Encryption Key can be directly generated onto the HSM.
- Oracle Pluggable database Master Encryption Key generated onto the HSM.

---

**NOTE:** Setup has 3 Oracle Homes which will be used for above described scenarios. Each Oracle Home will be used for each scenario

**Configuring the PKCS11 Provider on Oracle RAC Instances**

To set up SafeNet 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 SafeNet Luna HSM PKCS#11 library to the specified directory structure recommended by Oracle. Use the following directory structures.

- **UNIX**: 
  
  "/opt/oracle/extapi/[32,64]/hsm/{VENDOR}/{VERSION}/libapiname.ext"

- **Windows**: 
  
  "%SYSTEM_DRIVE%\oracle\extapi\[32,64]\hsm\{VENDOR\}\{VERSION\}\libapiname.ext"
### Integrating SafeNet HSM with Oracle Database 12c RAC

#### Parameter Definition

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>[32,64]</td>
<td>Specifies whether the supplied binary is 32-bits or 64-bits.</td>
</tr>
<tr>
<td>[vendor]</td>
<td>Specifies the name of the vendor supplying the library.</td>
</tr>
<tr>
<td>[version]</td>
<td>Refers to the version number of the library. <strong>Note:</strong> 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.</td>
</tr>
<tr>
<td>.ext</td>
<td>The extension must be replaced by the extension of the library file.</td>
</tr>
</tbody>
</table>

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/extapi/64/hsm/safenet/7.x.x/
# 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.

1. 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 /etc/oracle/wallet/
# chmod -R 700 /etc/oracle/wallet/
```

To test TDE with HSM, perform the following steps:
Verify that the 'traditional' software-based wallet is working properly

1. Add the following to your "$ORACLE_HOME/network/admin/sqlnet.ora" file on all instances of RAC (for e.g. RAC1, RAC2 and RAC3):

   ```
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/ORCL1RAC)))
   ```

2. Start the database:

   ```
   $ sqlplus / as sysdba
   ```

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 'keystore_location' IDENTIFIED BY software_keystore_password;
   ```

   'keystore_location' is the path to oracle wallet directory that was set in the sqlnet.ora file and 'software_keystore_password' must have length more than or equal to 8 characters.

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

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

   Run the above commands (2-7, if storing the wallet on shared disk only 2-5) on all instances RAC1, RAC2, and RAC3.

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. 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/

Above step is not needed if wallet is stored on shared location.

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. The next command lists encrypted columns in your database:
   SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;

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

Now connect to RAC 2 and RAC3 machine and execute the commands below after opening the wallet on machine.

13. Open the keystore.
   
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY software_keystore_password;

14. Now try to access the contents of EMPLOYEE table.
   
   SQL> SELECT * FROM EMPLOYEE;
   
   SQL> exit

Now when you start the database you need to open the wallet to view the encrypted data. You can set the wallet to Auto-Login that will automatically open when database starts

---

NOTE: Above commands works on all 3 nodes RAC1, RAC2 and RAC3 after copying the wallet on all instances.

---

Test if the database can reach the HSM device

1. Change your "$ORACLE_HOME/network/admin/sqlnet.ora" – file:

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

   For example:

   Change the FILE to HSM in your "$ORACLE_HOME/network/admin/sqlnet.ora" – file on all instances of RAC
   (for e.g. RAC1, RAC2 and RAC3):

   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/ORCL1RAC)))

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 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.

---

5. 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. 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;

7. 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';

8. 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.

9. Close the keystore:

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

10. Open the keystore:

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

11. 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 3 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.

12. 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) 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 'keystore_location'
   IDENTIFIED BY software_keystore_password;

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

13. 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.

14. Restart the database and 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

At this point when the database restarts the software wallet is opened automatically, but the HSM based wallet needs to be opened manually. To create the HSM auto wallet perform the following steps:

1. Change the sqlnet.ora entries as follows on all nodes:
   ```
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/ORCL1RAC)))
   ```
2. Rename the cwallet.sso as it is already present at keystore location. Rename it on all nodes.
3. Rename the encryption wallet as ewallet.p12 on all nodes.
4. Restart the database and connect as a system and 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 '/etc/oracle/wallet/ORCL1RAC' IDENTIFIED BY software_keystore_password;
   ```
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. Update the sqlnet.ora file to use the hardware security module on all nodes.
   ```
   SQL> ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/ORCL1RAC)))
   ```
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 Master Encryption Key directly on the HSM

In order to generate a Master Encryption Key for HSM-Based Encryption, perform the following instructions:

---

NOTE: It is assumed that no software or HSM based wallet is yet created.
1. Add the following to your "$ORACLE_HOME/network/admin/sqlnet.ora" file:

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

For example:

Create or add the following to your "$ORACLE_HOME/network/admin/sqlnet.ora" file on all instances of RAC (for e.g. RAC1, RAC2, and RAC3):

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

2. Restart the database and if the database is not yet started, you can start it using:

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

Check the status when database start command completes:

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

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, below is the snapshot of HSM partition contents:

   ```
   Partition Name: ORCL2
   Partition SN:    152042028
   Storage (Bytes): Total=102701, Used=1848, Free=100853
   Number objects:  5
   ```
To verify the master encryption key is encrypting the RAC configuration

1. Create a CUSTOMERS table in the database.
   ```sql
   SQL> CREATE TABLE CUSTOMERS (ID NUMBER(5), NAME VARCHAR(42), CREDIT_LIMIT NUMBER(10));
   ```
2. Enter some values in the CUSTOMERS table.
   ```sql
   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
   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
   SQL> SELECT CREDIT_LIMIT FROM CUSTOMERS;
   ```
5. The next command lists encrypted columns in your database:
   ```sql
   SQL> SELECT * FROM DBA_ENCRYPTED_COLUMNS;
   ```
6. Finally, this view contains information about the software keystore itself:
   ```sql
   SQL> SELECT * FROM GV$ENCRYPTION_WALLET;
   ```
7. Create an encrypted tablespace:
   ```sql
   SQL> CREATE TABLESPACE SECURESPACE DATAFILE '+DATA' SIZE 150M ENCRYPTION DEFAULT STORAGE (ENCRYPT);
   ```
8. Create a table in the tablespace:
   ```sql
   SQL> CREATE TABLE EMPLOYEE (ID NUMBER(5), NAME VARCHAR(42), SALARY NUMBER(10)) TABLESPACE SECURESACE;
   ```
9. Insert some values in EMPLOYEE table:
   ```sql
   SQL> INSERT INTO EMPLOYEE VALUES (001, 'JOHN SMITH', 15000);
   ```
3 – Integrating SafeNet HSM with Oracle Database 12c RAC

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;
   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

13. Now go to RAC2 and RAC3 machine, open the wallet on the node.
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "hsm_partition_password";

14. Now try to access the contents of EMPLOYEE table
   SQL> select * from employee;

Now onwards 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 that will automatically open when 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. Change the sqlnet.ora entries as follows on all RAC instances (RAC1, RAC2 and RAC3):
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/ORCL2RAC))

3. Create the software keystore in the appropriate location (for example, "/etc/oracle/wallet/ORCL2RAC1").
   SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/oracle/wallet/ORCL2RAC' 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 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 '/etc/oracle/wallet/ORCL2RAC' IDENTIFIED BY software_keystore_password;

8. Rename the ewallet.p12 to ewallet.p24 so that Oracle picks only cwallet.sso when auto login is enabled.
9. Update the sqlnet.ora file to use the hardware security module on all instances.
   
   `ENCRIPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/ORCL2RAC)))`

10. 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/
   ```

   **NOTE:** Above step is not required if you are using the shared storage to create the software wallet, which is accessible by all RAC nodes.

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.

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 ‘Integrating SafeNet HSM with Oracle Database 12c‘ for detailed steps.

Ensure that the changes corresponding to tnsnames.ora file are made on all RAC instances (RAC1, RAC2 and RAC3).
Integrating SafeNet HSM with Oracle Data Guard Physical Standby

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 sections describes the steps to take 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,

- Luna Client 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.
- SafeNet HSM PKCS#11 library is copied to the specified 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]  /opt/oracle/extapi/[32,64]/hsm/[VENDOR]/[VERSION]/libapiname.ext
[Windows] %SYSTEM_DRIVE%\oracle\extapi\[32,64]\hsm\[VENDOR]\[VERSION]\libapiname.ext

For example:

/opt/oracle/extapi/64/hsm/safenet/<7.x.x>/libCryptoki2_64.so
Scenario 1: Master key Migrated from Software Wallet 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 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 will return employee information.

5. 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 it shows any error saying "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 it shows an error saying “ORA-28374: typed master key not found in wallet”, 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.

- Ensure you follow the prerequisites listed in the section “Prerequisites”.

**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. To resume managed recovery, do the following

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

   Till this point we have provided all information about HSM wallet. When standby database recovery is completed, it can be opened in read-only mode to allow query access.

3. To switch the standby database into read-only mode, do the following

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

4. Now connect the database as system and execute 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

Steps to follow on Primary Database

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 secure01 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 secure01;

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 then copy the sqlnet.ora file from Primary Database to Standby Database at the following location on standby:

   $ORACLE_HOME/network/admin/sqlnet.ora

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

If you want to create the HSM Auto wallet on Primary and Standby database, follow the below steps on Primary 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 it shows any error saying “ORA-28374: typed master key not found in wallet”, 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;
```

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

**Steps to follow on Standby Database**

It is assumed that Standby database is configured.

- Ensure you follow the prerequisites listed in the section “Prerequisites”.
- If you created the auto wallet on 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.
- 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>)))
```

Perform the following steps to manage database recovery:

1. Connect to the database as sysdba.
   
   ```
   $ sqlplus / as sysdba
   ```

2. To resume managed recovery, do the following,
   
   ```
   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.

3. To switch the standby database into read-only mode, do the following,
   
   ```
   SQL> SHUTDOWN IMMEDIATE;
   SQL> STARTUP MOUNT;
   SQL> ALTER DATABASE OPEN READ ONLY;
   ```

4. 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 table spaces in standby.

```sql
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "hsm_partition_pwd";
```
5 – Troubleshooting Tips

Troubleshooting Tips

Problem – 1

When trying to generate the master key on the HSM or migrate keys from the wallet to the HSM:

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 trying to open the keystore or generate the Master Key on the HSM.

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, if it enabled then please use 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 is closing when using the Oracle (12.1.0.1 / 12.1.0.2) on Windows and Solaris and oracle alert.log is showing the 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