Oracle Database with PSE

Integration Guide

gemalto

security to be free

Document Number: 007-012150-001, Rev. B

Release Date: May 2018

Contents

Preface

This document is intended to guide security administrators through the steps for the Oracle Database and Protect Server External (PSE) integration.

Scope

This document outlines the steps to integrate Oracle Database with PSE. PSE is used to secure the Master Encryption Key for Oracle TDE in FIPS 140-2 Approved HSM.

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

Convention	Description
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

Contact Method	**Contact Information**	
Address	Gemalto 4690 Millennium Drive Belcamp, Maryland 21017, USA	
Phone	US	1-800-545-6608
	International	1-410-931-7520
Technical Support Customer Portal	https://supportportal.gemalto.com Existing customers with a Technical Support Customer Portal account can log in to manage incidents, get the latest software upgrades, and access the Gemalto Knowledge Base.	

1 Introduction

Overview

This guide provides the necessary information to install, configure and integrate Oracle Database Transparent Data Encryption (TDE) with SafeNet PSE Hardware Security Module (HSM).

TDE provides the infrastructure necessary for implementing encryption. It enables to encrypt sensitive data stored in application table columns (such as credit card numbers etc.) or application table spaces, the containers for all objects stored in a database TDE prevents data theft of confidential data stored on media. The motivation for the Oracle TDE to use the PSE HSM for EKM is because of the following reasons:

- It is used to store the master encryption keys used for transparent data encryption. And the master encryption key is never exposed in insecure memory.
- It also provides more secure computational storage.
- PSE is a more secure alternative to the Oracle wallet.

3rd Party Application Details

- Oracle Database 12c R2
- Oracle Database 11g R2

Supported Platforms

The following platforms are tested with PSE:

Oracle Database 12c R2 version: 12.2.0.1

Platforms Tested	Firmware Version	Distribution
Solaris Sparc 11 64-bit	5.02.00	PSE 5.5.0

Oracle Database 11g R2 version: 11.2.0.3

Platforms Tested	Firmware Version	Distribution
Aix 6.1 64-bit	3.20.01	PSE v4.2.0 PSE v4.2.1

Prerequisites

Protect Server External Setup:

- Set AC Switch to 230v.
- Connect PS/2 Keyboard.
- Connect VGA screen.
- Connect power cable.
- At login prompt, PSE version was reported as 1.1.0 DOM.
- Logged in as root user, password = password.
- Run passwd command to change the root password to the company usual.
- Run hsmstate which responded with NORMAL MODE. RESPONDING.
- Run cat /etc/sysconfig/network-scripts/ifcfg-eth0.
- Open the file /etc/sysconfig/network-scripts/ifcfg-eth0 to change the IP Address of the PSE according to your network configuration.
- Restart the network using /etc/init.d/network restart.
- Run iptables -L to find that all chains had their default policy set to ACCEPT.
- Run cat /etc/issue which contains Protect Server External 1.1.0 DOM.

ProtectToolkit C installation on Solaris Sparc

- Install SafeNet Network HSM Access Provider 5.5.0.

- Install SafeNet ProtectToolkit C SDK 5.5.0.

By default PTK is installed in S/W mode to change it in HSM mode following steps needs to be followed:

- Go to `/etc/defult/` open file `et_hsm` modify it to include IP address of HSM, by default loopback address is configured in this file.
- Create following soft link:

```
ln -s /opt/safenet/protecttoolkit5/cpsdk/lib/sparc/sparcv9/libcthsm.so
/opt/safenet/protecttoolkit5/ptk/lib/sparcv9/libcryptoki.so
```

- Now execute `hsmstate`, it displays HSM info.

```
HSM device 0:   HSM in NORMAL MODE. RESPONDING. Usage Level=0%
```

- Initialize the HSM using `ctkmu`.

```
root@hsm-solaris-sparc11:/opt/safenet/protecttoolkit5/ptk/bin# ctkmu t -s6
ProtectToolkit C Key Management Utility 5.5.0
Copyright (c) Safenet, Inc. 2009-2017

Enter the new token label: Oracle

Please enter Security Officer PIN for the new token in Slot 6:
Please confirm Security Officer PIN for the new token in Slot 6:
Initializing token in Slot 6:
Setting user pin for new token in slot 6:
Please enter User's PIN for the new token in slot 6:
Please confirm User's PIN for the new token in slot 6:
```

NOTE: Enter token label and set the PIN for the respective slot which will be later used for this integration.

```
root@hsm-solaris-sparc11:/opt/safenet/protecttoolkit5/ptk/bin# ctkmu l
ProtectToolkit C Key Management Utility 5.5.0
Copyright (c) Safenet, Inc. 2009-2017
Cryptoki Version  = 2.20
Manufacturer      = Safenet, Inc.
test                              (Slot 0)
Aamir                             (Slot 1)
MACHINE_Keyset                    (Slot 2)
SYSTEM_Keyset                     (Slot 3)
DEVSHARM_Keyset                   (Slot 4)
test                              (Slot 5)
Oracle                            (Slot 6)
<uninitialised token>             (Slot 7)
<uninitialised token>             (Slot 8)
<uninitialised token>             (Slot 9)
AdminToken (527012)               (Slot 10)
root@hsm-solaris-sparc11:/opt/safenet/protecttoolkit5/ptk/bin#

root@hsm-solaris-sparc11:/opt/safenet/protecttoolkit5/ptk/bin# ctconf -ft
ProtectToolkit C Configuration Utility 5.5.0
Copyright (c) Safenet, Inc. 2009-2017
At least 1 other application is still running.
Security Mode has not been set.
root@hsm-solaris-sparc11:/opt/safenet/protecttoolkit5/ptk/bin#
```

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 providing the access of **`libcthsm.so`** and **`libetnetclient.so`**.

Providing access of library for Non-Root user

To allow non-root users or applications access to the HSM, change the ownership of **`libcthsm.so`** and **`libetnetclient.so`** and change the permission of respective library.

Changing the ownership and permission of library

Ensure that you have **sudo** privileges on the client workstation.

a. Changing the ownership of library for Non-root user

```
chown -R oracle:oinstall /opt/safenet/protecttoolkit5/cpsdk/lib/sparc/sparcv9/libcthsm.so
```

```
chown -R oracle:oinstall
/opt/safenet/protecttoolkit5/nethsm/lib/sparc/sparcv9/libetnetclient.so
```

b. Changing the permission of library for Non-root user

```
chown -R oracle:oinstall /opt/safenet/protecttoolkit5/cpsdk/lib/sparc/sparcv9/libcthsm.so
```

```
chmod -R 755 /opt/safenet/protecttoolkit5/cpsdk/lib/sparc/sparcv9/libcthsm.so
```

```
chown -R oracle:oinstall
/opt/safenet/protecttoolkit5/nethsm/lib/sparc/sparcv9/libetnetclient.so
```

```
chmod -R 755 /opt/safenet/protecttoolkit5/nethsm/lib/sparc/sparcv9/libetnetclient.so
```

Oracle Database Setup

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 PSE with Oracle Database 12c

Setting up PSE for Transparent Data Encryption

To set up PSE for Transparent Data Encryption, perform the following steps:

Create a soft Link PSE PKCS#11 library to the specified directory structure to ensure that the oracle database is able to find this library. Use the following directory structure:

`"/opt/oracle/extapi/[32,64]/hsm/{Vendor}/{Version}/"` (Solaris Sparc)

For example,
/opt/oracle/extapi/64/hsm/safenet/5.5.0/

Where:

- [32, 64] specifies whether the supplied binary is 32-bits or 64-bits.
- Vendor stands for the name of the vendor supplying the library.
- Version refers to the version of the library. This should preferably be in the below format:

  number.number.number

  The API name requires no special format. However, the XX must be prefixed with the word lib, as illustrated in the syntax. The extension, ext needs to be replaced by the extension of the library file.

After creating above directory structure create following soft link:

```
ln -s /opt/safenet/protecttoolkit5/cpsdk/lib/sparc/sparcv9/libcthsm.so
/opt/oracle/extapi/64/hsm/safenet/5.5.0/libcryptoki.so
```

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

Oracle user should have the read/write permission of the above directory and file.

Login as an Oracle user to export the following variables:

```
export ORACLE_SID=orcl
export ORACLE_BASE=/u01/app/oracle (oracle installation directory)
export ORACLE_HOME=$ORACLE_BASE/product/12.2.0/dbhome_1
export PATH=$PATH:$ORACLE_HOME/bin
export TNS_ADMIN=$ORACLE_HOME/network/admin
```

Generating a Master Encryption Key for HSM-Based Encryption

To start using HSM-based encryption, you need to have a master encryption key that will be stored inside the HSM. The master encryption key is used to encrypt or decrypt column/tablespace encryption keys inside the HSM. 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

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

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 PSE, perform the following steps:

Verify that the 'traditional' software-based wallet is working fine:

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 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 "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;
   ```

NOTE: 'keystore_location' is the path to oracle wallet directory that you 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;
   ```

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. Optionally, you can 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. Create a CUSTOMERS table in the database.

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

9. 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);
   ```

10. Encrypt the 'CREDIT_LIMIT' column of the 'CUSTOMERS' table:

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

11. 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;
    ```

12. The following command lists encrypted columns in your database:

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

13. The keystore view contains information about the software keystore:

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

14. Create an encrypted tablespace:

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

15. Create a table in the tablespace:

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

16. 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);
```

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

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

18. Close the keystore:

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

19. 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

20. Open the keystore:

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

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>)))
    ```

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 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. The software_keystore_password is the password given the software wallet in step 1.

4. With the following command, the values listed in the encrypted column are returned in clear text. Transparent Data Encryption decrypts them automatically using the HSM master key:

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

5. Change the password of software keystore to 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';
    ```

    From now onwards when you open the keystore, it will open both software-based keystore as well as 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 software based wallet (if you want to do) 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;
   ```

   To use the auto-login wallet only on local system use 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 to ensure that oracle uses auto-open wallet.

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

11. Restart the database and connect to the database as system and open the HSM keystore(software wallet will open automatically):

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

Create HSM Auto Wallet when HSM and Auto-Open Software wallet is 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 a 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. Check the wallet information with the following command:

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

Generating Master Encryption Key directly onto 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.

Setting up Oracle to create Master Encryption Key onto 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 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 "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 HSM, below is the snapshot of HSM partition contents:

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

ORACLE.TSE.HSM.MK.072AC159D9153C4FF0BF3BF931ED9693850203 – SECRET_KEY       AES
ORACLE.SECURITY.KM.ENCRYPTION.303636313238364138433731383634463241424637383931443034343135 3
4443941 – DATA    RSA

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

7. Create a CUSTOMERS table in the database.

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

8. 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);
```

9. Encrypt the 'CREDIT_LIMIT' column of the 'CUSTOMERS' table:

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

10. 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;
```

11. The following command lists encrypted columns in your database:

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

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

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

13. Create an encrypted tablespace:

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

14. Create a table in the tablespace:

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

15. 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);
```

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

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

17. Close the keystore:

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

18. 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
```

19. Open the keystore:

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

Create HSM Auto Wallet

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

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

21. Change the `sqlnet.ora` entries as follows:

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

22. 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;
```

23. Open the software keystore:

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

24. 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';
```

25. Close the software keystore:

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

26. Create Auto-Login keystore.

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

27. 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.

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

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

Working with Pluggable Databases (PDB)

A new feature for Oracle Database 12c is Multitenant Architecture, Oracle Multitenant delivers a new architecture that allows a multitenant container database to hold many pluggable databases. The 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.

Managing Pluggable Databases

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 the most preferred one is to use DBCA utility. It is assumed that you have already created PDBs. For demonstrated purpose in this guide we are using the PDB with named "salespdb".

TDE in Pluggable Databases

Below are the steps to use TDE with Pluggable Databses:

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

   For Example:

   ```
   salespdb =

   (DESCRIPTION =
          (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
           (CONNECT_DATA =
             (SERVER = DEDICATED)
             (SERVICE_NAME = salespdb.localdomain)
          )
     )
   ```
   Where, **salespdb** is the new Pluggable database name.

2. Restart the Listener Service.

   ```
   # lsnrctl stop

   # lsnrctl start
   ```

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. Run the below grant commands to PDB Admin:

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

   Grant succeeded.

   ```
   SQL> GRANT CREATE SESSION TO salesadm;
   ```

   Grant succeeded.

   ```
   SQL> GRANT CONNECT TO salesadm;
   ```

   Grant succeeded.

   ```
   SQL> GRANT DBA TO salesadm;
   ```

   Grant succeeded.

   ```
   SQL> GRANT CREATE ANY TABLE TO salesadm;
   ```

   Grant succeeded.

   ```
   SQL> GRANT UNLIMITED TABLESPACE TO salesadm;
   ```

Grant succeeded.

```
SQL> ALTER USER salesadm PROFILE DEFAULT;
```

User altered.

```
SQL> commit;
```

Commit complete.

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

5. Try connecting to PDB with PDB username and you should be able to connect it:

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

   For Example:

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

Generating 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
   Connected
   ```

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

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

   For example:

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

   NOTE: Please make sure that keystore for CDB (root container) is opened and Master key for CDB is generated before opening the keystore and generating the Master key for PDB.Also do not configure HSM auto login for CDB until you generate the master key for PDB (all PDB in case multiple PDB 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 as well.

4. Run the following SQL statement to create the PDB Master key:

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

   For example:

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "userpin2";
   keystore altered.
   ```

NOTE: This will generate a new master key and from now onwards any encryption/decryption operations performed within this pdb will use this master key.

5. Create a CUSTOMERS table in the PDB.

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

6. 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);
   ```

7. Encrypt the 'CREDIT_LIMIT' column of the 'CUSTOMERS' table:

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

8. 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;
   ```

9. The following command lists encrypted columns in your databases:

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

10. Create an encrypted tablespace:

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

11. Create a table in the tablespace:

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

12. 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);
    ```

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

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

14. Close the keystore:

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

15. 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
    ```

16. Open the keystore:

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

The above queries are same as we did for root container but it will use the Master Encryption Key of PDB generated on HSM.

3 Integrating PSE with Oracle Database 11g R2

Setting up PSE for Transparent Data Encryption

To set up PSE for Transparent Data Encryption, perform the following steps:

Create below directory structure to ensure that the database is able to find PKCS#11library.

```
"/opt/oracle/extapi/[32,64]/hsm/{Vendor}/{Version}/"
```

For example,
`/opt/oracle/extapi/64/hsm/safenet/4.2.1/`

Where:

- [32, 64] specifies whether the supplied binary is 32-bits or 64-bits.
- Vendor stands for the name of the vendor supplying the library.
- Version refers to the version of the library. This should preferably be in the below format:

  number.number.number

  The API name requires no special format. However, the XX must be prefixed with the word lib, as illustrated in the syntax. The extension, ext needs to be replaced by the extension of the library file.

After creating above directory structure create following soft link:

```
ln -s /opt/PTK/lib/aix-ppc64/libcthsm.a /opt/oracle/extapi/64/hsm/safenet/4.2.1/libcryptoki.so
```

Assign following ownership and permissions:

```
chown oracle:dba /opt/ETcprt/lib/aix-ppc/aix-ppc64/libcthsm.a
chmod 755 /opt/ETcprt/lib/aix-ppc/aix-ppc64/libcthsm.a
chown oracle:dba /opt/ETnethsm/lib/aix-ppc/aix-ppc64/libetnetclient.a
chmod 755 /opt/ETnethsm/lib/aix-ppc/aix-ppc64/libetnetclient.a
```

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

Oracle user should have the read/write permission of the above directory and file and after logged on as oracle you need to export the following variables:

```
export ORACLE_SID=orcl
export ORACLE_BASE=/opt/app/oracle/ (oracle installation directory)
export ORACLE_HOME=$ORACLE_BASE/product/11.2.0/dbhome_1/
export PATH=$PATH:$ORACLE_HOME/bin
export TNS_ADMIN=$ORACLE_HOME/network/admin
```

Generating a Master Encryption Key for HSM-Based Encryption

To start using HSM-based encryption, you need to have a master encryption key that will be stored inside the HSM. The master encryption key is used to encrypt or decrypt column/tablespace encryption keys inside the HSM. 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.
- Automatic wallet management across Oracle RAC instances.

Migrating Master Encryption Key onto the HSM

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

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 PSE, perform the following steps:

Verify that the 'traditional' software-based wallet is working fine:

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 sysoper
```

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

```
SQL> startup;
```

3. 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 "temp123#".

4. Create an encryption wallet; the master key is added into it automatically; the double quotes are mandatory:

```
SQL> alter system set encryption key identified by "wallet_password";
```

NOTE: “wallet_password” must contain alphanumeric characters and have length more than or equal to 8 characters.

5. Encrypt the 'credit_limit' column of the 'CUSTOMERS' table which is owned by the user 'OE':

   ```
   SQL> alter table oe.customers modify (credit_limit encrypt);
   ```

6. 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 oe.customers where rownum <15;
   ```

7. The following command lists encrypted columns in your database:

   ```
   SQL> select * from dba_encrypted_columns;
   ```

8. Finally, this view contains information about the wallet itself:

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

9. Create an encrypted tablespace:

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

10. Close the wallet:

    ```
    SQL> alter system set wallet close identified by “wallet_password”;
    SQL> exit
    ```

Test if the database can reach the HSM device:

11. Change your $ORACLE_HOME\network\admin\sqlnet.ora – file:

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

12. $ sqlplus

    Connect to the database as 'system':

13. Migrate the wallet onto the HSM device:

    ```
    SQL> alter system set encryption key identified by “hsm_partition_pwd |<slot_lable>” migrate
    using “wallet_password;
    ```

    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. The software_keystore_password is the password given the software wallet in step 1.

14. 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 credit_limit from oe.customers where rownum <15;
    ```

15. Close the wallet:

    ```
    SQL> alter system set wallet close identified by “wallet_password”;
    SQL> exit
    ```

16. Start Oracle Wallet Manager from Start Menu:
17. Open the software-based wallet and click on 'Change Password'; use the same string you used for the HSM wallet as the new password for the software based wallet in the form “hsm_partition_pwd”; click on “Save”, then “Exit”.
18. $ sqlplus

    Connect to the database as 'system':

```
SQL> alter system set wallet open identified by "hsm_partition_pwd"
```

This opens both the HSM and the software wallet.

19. Close the wallet:

```
SQL> alter system set wallet close identified by "wallet_password";
SQL> exit
```

20. Start Oracle Wallet Manager from Start Menu:
21. Open the software-based wallet, change the password back to the initial password, check 'Auto-Login'; click on 'Save', then 'Exit'
22. Verify that an auto-open software wallet 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 encryption wallet:

    $ < path to the oracle wallet directory >rename ewallet.p12 ewallet.p24

    so that Transparent Data Encryption does not try to open it.
23. Connect to the database as system and open the HSM wallet (the software is already open):

```
SQL> alter system set wallet open identified by "hsm_partition_pwd";
```

Generating Master Encryption Key directly onto 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.

Setting up Oracle to create Master Encryption Key onto 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 sysoper
```

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

```
SQL> startup;
```

3. 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 "temp123#".

4. Create an encryption wallet. The master key would automatically be created onto the HSM.

```
SQL> alter system set encryption key identified by "hsm_partition_pwd";
```

To use multiple slots or partitions, the syntax to generate master key to a slot or partition, use the following syntax:

```
SQL> alter system set encryption key identified by "hsm_partition_pwd|<slot_name>";
```

5. Encrypt the 'credit_limit' column of the 'CUSTOMERS' table which is owned by the user 'OE':

```
SQL> alter table oe.customers modify (credit_limit encrypt);
```

6. 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 oe.customers where rownum <15;
```

7. The following command lists encrypted columns in your database:

```
SQL> select * from dba_encrypted_columns;
```

8. Finally, this view contains information about the wallet itself:

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

9. Create an encrypted tablespace:

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

10. Close the wallet:

```
SQL> alter system set wallet close identified by “wallet_password”;

SQL> exit
```

4
Troubleshooting Tips

Problem – 1

Error message "ORA-43000: PKCS11: library not found" or "ORA-28376: cannot find PKCS11 library" when trying to generate Master Key on HSM or migrate keys from wallet to HSM.

Solution:

1. Ensure that library path is set correctly, for example:

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

2. Ensure that oracle:oinstall is the owner: group of this directory with read/write permissions.
3. Ensure that 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.