Oracle

Oracle TDE

23c/23ai

Integration Guide

SecurityServer

6.1.1

utimaco®

Imprint

Copyright 2025	Utimaco IS GmbH Germanusstr. 4 D-52080 Aachen Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226) EMEA +49 800-627-3081 APAC +81 800-919-1301
Internet	https://support.hsm.utimaco.com/
e-mail	support@utimaco.com
Document Version	1.0.0
Date	2025-08-25
Status	PUBLISHED
Document No.	IG-2025-0049

All rights reserved	No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems. Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them. All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco to facilitate secure database encryption practices. It outlines the integration of Oracle Transparent Data Encryption (TDE) with Utimaco's Hardware Security Module (HSM), enabling robust key management and enhanced data protection.

1.1 About This Guide

This guide describes how to integrate Oracle Transparent Data Encryption (TDE) with Utimaco HSM to enable secure key management for encrypted database operations. The primary objectives of this integration are:

- Strengthen data protection by securely managing encryption keys within the Utimaco HSM.
- Enable Oracle TDE functionality using external key management for compliance and enhanced security.

This guide walks through the required configuration steps, including setting up a key store, managing credentials, and encrypting Oracle database operations using TDE with HSM.

1.2 Target Audience

This guide is intended for Oracle database administrators and Utimaco HSM administrators.

1.3 Purpose of the Integration

Integrating Oracle Transparent Data Encryption (TDE) with Utimaco Hardware Security Module (HSM) establishes a secure and centralized key management solution for protecting sensitive data stored in Oracle databases. This integration ensures that encryption keys are generated, stored, and managed within a certified hardware-based environment, significantly reducing the risk of unauthorized access or data breaches.

1.4 Abbreviations

Abbreviation	Meaning
HSM	Hardware Security Module
PKI	Public Key Infrastructure
TDE	Transparent Data Encryption
PKCS	Public Key Cryptography Standards
PKCS#11	PKCS Part 11: The Cryptographic Token Interface Standard
SO	The PKCS#11 cryptographic slot Security Officer
DB	Database
JRE	Java Runtime Environment
MBK	Master backup key
OCI	Oracle Cloud Infrastructure
KMS	Key Management Service
SID	System Identifier
LAN	Local Area Network.
RHEL	Red Hat Enterprise Linux

API	Application Programming Interface
IP	Internet Protocol
PCIe	Peripheral Component Interconnect Express
MEK	Master Encryption Key
SQL	Structured Query Language
PDB	Pluggable Database
CDB	Container Database
SYS	System User (in Oracle Database, SYS is the default DBA user)
DBCA	Database Configuration Assistant
SYSDBA	System Database Administrator (a privileged Oracle role)
SSO	Single Sign-On

Table 1: Abbreviations

1.5 Document Conventions

The following conventions are used in this guide:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Click **New Device**

Convention	Use	Example
`Monospaced`	Code that is given for explanation or as an example, file paths	`takeown /F "C:\oracle\extapi\64\hsm\utimaco\6.1.1" /A`
Italic	References and important terms	See *Oracle database documentation link* in the Setting Up Oracle TDE

Table 2: Document Conventions

We use special icons to highlight the most important notes and information.

Here you will find important safety information that should be followed.

Here you will find additional notes or supplementary information.

This message marks the result expected after the successful execution of an instruction.

2 Product Overview

2.1 Overview of Oracle TDE

Oracle Database uses authentication, authorization, and auditing mechanisms to secure data in the database, but not in the operating system data files where data is stored. To protect these data files, Oracle Database provides Transparent Data Encryption (TDE). TDE encrypts sensitive data stored in data files. To prevent unauthorized decryption, TDE stores the encryption keys in a security module external to the database. This security module can be referred to as follows:

- TDE wallets are wallets used for TDE. They cannot contain other security artifacts such as certificates. In previous releases, they were called software keystores or just wallets.
- External keystores refer to Oracle Key Vault or Oracle Cloud Infrastructure (OCI) Key Management Service (KMS).
- Keystores is a generic term for both TDE wallets and external keystores.

Transparent Data Encryption (TDE) enables you to encrypt sensitive data stored in tables and tablespaces and database backups.

After the data is encrypted, it is transparently decrypted for authorized users or applications when they access it. TDE helps protect data stored on media (also called data at rest) if the storage media or data file is stolen.

2.2 Overview of Utimaco CryptoServer HSM

CryptoServer is a hardware security module developed by Utimaco IS GmbH. It is a physically protected, specialized computer unit designed to perform sensitive cryptographic tasks and securely manage and store cryptographic keys and data. CryptoServer can be used as a universal, independent security component for heterogeneous computer systems.

2.3 Joint Value Proposition

Integrating Oracle Transparent Data Encryption (TDE) with Utimaco SecurityServer (CryptoServer HSM) offers a robust and compliant solution for securing sensitive data at rest. Oracle TDE provides seamless encryption of database tablespaces and columns. At the same time, Utimaco SecurityServer ensures that the master encryption keys are stored in a tamper-proof hardware security module, enhancing protection against unauthorized access and insider threats. This setup uses PKCS#11 to manage encryption keys securely and centrally.

3 Integration Requirements and Prerequisites

Ensure that the system environment you will be using meets the following hardware and software requirements.
This guide assumes that the user has already installed and configured an Oracle database.

3.1 Tested Versions

This guide assumes that the user has already installed and configured Oracle database.

Operating System	Oracle TDE Version	Utimaco Security Server Version	Utimaco HSM
Windows Server 2022	23c/23ai, 21c	6.1.1	CryptoServer CSe-Series/Se-Series
RHEL 9.4	23c/23ai, 19c	6.1.1	CryptoServer CSe-Series/Se-Series
Oracle Linux 9	23c/23ai, 19c	6.1.1	CryptoServer CSe-Series/Se-Series

Table 3: List of Tested Versions

In Windows Server 2022 Oracle 19c is not officially supported on Windows Server 2022 unless Patch 19.13 or later is applied post-installation using Oracle's patching tools.

Additionally, Oracle 12c is also not supported on Windows Server 2022. As per Oracle's documented compatibility, the last supported Windows Server versions for 12c are:

- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2

In RHEL 9.4, Oracle 21c is not officially certified, and Oracle Database 12c is also not certified for this release. The latest supported OS versions for Oracle Database 12c, specifically Release 12.2, are RHEL 7.x and RHEL 8.2. Additionally, in Oracle Linux 9, both Oracle 21c and 12c are not officially certified.

3.2 Hardware and Software Requirements

3.2.1 Hardware Requirements

Hardware	Hardware Requirements
Utimaco LAN HSM	CryptoServer CSe-Series/Se-Series LAN with firmware SecurityServer 6.1.1 or higher
Utimaco PCI-e HSM	CryptoServer CSe-Series/Se-Series PCI-e with firmware SecurityServer 6.1.1 or higher

Table 4: List of Hardware Requirements

3.2.2 Software Requirements

Software	Software Requirements
Oracle	Oracle 23ai, 21c, 19c
JRE 8	Java Runtime Environment 8
HSM Interface	SecurityServer PKCS#11 Provider

Table 5: List of Software Requirements

3.3 Prerequisites

Before you begin, please ensure that you have installed/setup:

- Operating system listed in Tested Versions.
- SecurityServer listed in Tested Versions.
- CryptoServer Default Admin should be replaced with a new admin user.
- MBK must be created and stored onto each HSM. Refer the CryptoServer documentations to setup the MBK.
- CryptoServer is setup and configured. Refer the CryptoServer documentations to setup the HSM.
- The PKCS#11 library is set up and configured according to your environment. Refer to the CryptoServer documentation to set up and configure the PKCS#11 library.
- The user with admin privileges is required to install a few packages on the Oracle Database server.

4 Installation and Configuration

4.1 Setting Up Utimaco SecurityServer Software

On Linux:

1. Copy the downloaded software to the appropriate location on the Oracle Database Server.
2. Create a utimaco folder under the /opt directory and create 2 directories /opt/utimaco/bin and /opt/utimaco/lib.

```
# mkdir -p /opt/utimaco/bin
# mkdir /opt/utimaco/lib
```

3. Copy the pkcs11 library file libcs_pkcs11_R3.so from the Utimaco CryptoServer software to the /opt/utimaco/lib directory and make the file executable.

```
cp ~/path_to_application_folder/lib/libcs_pkcs11_R3.so /opt/utimaco/lib
chmod +x /opt/utimaco/lib/libcs_pkcs11_R3.so
```

4. Copy the csadm and p11tool2 files from the Utimaco CryptoServer software to the /opt/utimaco/bin directory and make both files executable.

```
# cd ~/path_to_application_folder
# cp csadm p11tool2 /opt/utimaco/bin
# chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2
```

5. Create the directory /etc/utimaco. Locate the Utimaco PKCS#11 configuration file in your SecurityServer directory, Software\Linux\Crypto_APIs\PKCS11_R3\sample. Copy the Utimaco PKCS#11 configuration file cs_pkcs11_R3.cfg to /etc/utimaco directory.

```
# mkdir /etc/utimaco
# cd ~/path_to_application_folder/Software/Linux/Crypto_APIs/PKCS11_R3/sample
# cp cs_pkcs11_R3.cfg /etc/utimaco
```

On Windows:

On Windows, cs_pkcs11_R3.cfg will be automatically created and available in the C:\ProgramData\UtimacoPKCS11_R3 folder as part of the CryptoServer software installation.

Edit the cs_pkcs11_R3.cfg file and make the appropriate changes to the file.

cs_pkcs11_R3.cfg (Sample file)

```
library = C:\oracle\extapi\64\hsm\utimaco\6.1.1.0\cs_pkcs11_R3.dll
slot = 0
pin = Oracle123

[Global]

# For Unix:

Logpath = /tmp

# For Windows:

# Logpath = C:/ProgramData/Utimaco/PKCS11_R3

# Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)

Logging = 1

# Prevents expiring session after inactivity of 15 minutes

KeepAlive = true

# Set the Device to connect with

#[CryptoServer]

# Device specifier

Device = <HSM_IP>
```

For detailed guidance on commands and their parameters, please refer to the Utimaco CryptoServer documentation.

The device could be a CryptoServer HSM, available in either PCIe or LAN form factors. Depending on the type, the device configuration line will follow one of these formats:

- LAN-based HSM:
  `Device = 288@ipaddress`
- PCIe-based HSM:
  `Device = /dev/cs2.0`

Make sure to select the appropriate format based on your specific hardware setup.

`library` specifies the path where the cs_pkcs11_R3.dll file is located.

`Slot` indicates the slot number associated with the created USER.

`Pin` represents the password assigned to the USER.

To simplify your testing process, it's recommended that you enable the PKCS#11 log file by adjusting the logging settings. Specifically:

- Set the `LogPath` to a writable directory (not a specific file).
- Set the `Logging Loglevel` to 1 for basic logging. Increase it to 4 for more detailed output during testing.

This will generate a log file named `cs_pkcs11_R3.log` within the specified `LogPath` directory. Reviewing this log can help with troubleshooting if you encounter issues.

Once testing is complete, it's advisable to reduce `Logging Loglevel` to `1` or `2` to limit output to only critical or important messages.

4.2 Setting Up Oracle TDE

Oracle Database must be installed on the target machine to continue the integration process. For a detailed installation procedure, refer to the Oracle Database documentation according to the desired version of Oracle. For the Oracle TDE 23ai version, refer to *Oracle Database 23ai—Install*.

5 Integration Steps

5.1 Copying Utimaco PKCS#11 Library File

If your host server has a 32-bit or 64-bit architecture, ensure the corresponding directory exists; if not, create it manually.

On Linux:

Parameter	Definition
[32 bit]	$ORACLE_BASE/extapi/32/hsm[/hsm-manufacturer/library-version/]
[64 bit]	$ORACLE_BASE/extapi/64/hsm[/hsm-manufacturer/library-version/]

Table 6: List of Parameters and Definitions

Make ownership and permissions on the above directory as:
owner=oracle; group=oinstall; permissions=775.

On Windows:

Parameter	Definition
[32 bit]	C:\oracle\extapi\32\hsm[\hsm-manufacturera\library-version\]
[64 bit]	C:\oracle\extapi\64\hsm[\hsm-manufacturera\library-version\]

Table 7: List of Parameters and Definitions

Make sure the 'oracle' user can access the above Windows folder.

1. To set folder ownership, use Command Prompt (Admin).

```
takeown /F "C:\oracle\extapi\64\hsm\utimaco\6.1.1" /A
```

2. Then use ICACLS to assign ownership.

```
icacls "C:\oracle\extapi\64\hsm\utimaco\6.1.1" /setowner oracle
```

3. Set Permissions.

```
icacls "C:\oracle\extapi\64\hsm\utimaco\6.1.1" /grant oracle:(OI)(CI)F oinstall:
(OI)(CI)M Users:(OI)(CI)R
```

Valid directory examples on 64-bit are:

Parameter	Definition
Linux	/opt/oracle/extapi/64/hsm/utimaco/6.1.1/
Windows	C:\oracle\extapi\64\hsm\utimaco\6.1.1\

Table 8: List of Parameters and Definitions

Please copy the Utimaco PKCS#11 (cs_pkcs11_R3.dll or libcs_pkcs11_R3.so) library for your host architecture to the folder mentioned above to allow the Oracle database to access the cryptographic library.

Oracle Database should now be able to access the Utimaco PKCS#11 HSM provider.

5.2 Configuring Oracle DB to use Utimaco HSM

To use HSM-based encryption, a Master Encryption Key (MEK) must be generated and securely stored within the Hardware Security Module (HSM). This key is the root for encrypting and decrypting Oracle database columns and tablespaces.

The HSM architecture clearly separates general database operations and cryptographic functions. This separation enables role-based access control, allowing database and security administrators to manage their responsibilities independently. For example, the keystore

password can be withheld from the database administrator, requiring the security administrator to provide it when needed, thereby enhancing overall security.

Unlike software-based keystores, the HSM is a physical device that protects the MEK from unauthorized access. All cryptographic operations involving the MEK are executed within the secure boundaries of the HSM, ensuring that the key is never exposed to system memory or vulnerable software layers.

We have used Windows Server while performing this integration. The SQL commands use Windows-style paths; change the path according to the appropriate operating system.

5.3 Generating the Master Encryption Key (MEK) on to the HSM

1. Create a wallet directory in the Oracle base path, typically in C:\oracle\admin\<db_unique_name>\wallet.
2. Log in to the database instance as a user granted the SYSDBA administrative privilege.

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

3. Set the WALLET_ROOT parameter.

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

4. Shut down and start up the database.

```
SQL> shutdown immediate;
SQL> startup;
```

5. Set the TDE_CONFIGURATION parameter.

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

6. Grant the ADMINISTER KEY MANAGEMENT or SYSKM privilege to SYSTEM and any user you want.

```
SQL> grant ADMINISTER KEY MANAGEMENT to system;
SQL> commit;
```

7. Connect to the database as a system user.

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

8. Run the ADMINISTER KEY MANAGEMENT SQL statement to open the HSM-based keystore.

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

9. Set the MEK in the HSM keystore.

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

10. You can verify that the key gets generated on the HSM using the following command.

```
p11tool2 LoginUser=<hsm_password> ListObjects
```

Example:

```
C:\Program Files\Utimaco\SecurityServer\Administration>p11tool2.exe LoginUser=Oracle123 ListObjects


CKO_DATA:

+ 1.1
  CKA_UNIQUE_ID                 = A519F8C2-2479-413E-9A0F-AF6C0943E7CE
  CKA_LABEL                     = DATA_OBJECT_SUPPORTED_IDEN

+ 1.2
  CKA_UNIQUE_ID                 = 128D1839-BF0F-4467-BE06-CBF52A3C946D
  CKA_LABEL                     = ORACLE.SECURITY.KM.ENCRYPTION.303630313646313539334635413734463343424636333423545423
1413539333731


CKO_SECRET_KEY:

+ 2.1
  CKA_KEY_TYPE                  = CKK_AES
  CKA_UNIQUE_ID                 = A2537001-3C30-4809-A686-0F76EC86477C
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = ORACLE.TDE.HSM.MK.06016F1593F5A74F3CBF633B5EB1A59371
  CKA_ID                        =


C:\Program Files\Utimaco\SecurityServer\Administration>
```

Figure 1 : List Objects

6 Verification and Testing

6.1 Verifying Encrypted Columns in Tables

1. Create a SCIENTISTS table in the DB.

```
SQL> create table SCIENTISTS (SCID NUMBER(4), FirstName VARCHAR2(128), LastName
VARCHAR2(128), Salary NUMBER(6));
```

2. Add data to the SCIENTISTS table.

```
SQL> insert into SCIENTISTS values (0001, 'Albert', 'Einstein', 850000);
SQL> insert into SCIENTISTS values (0002, 'Isaac', 'Newton', 750000);
SQL> insert into SCIENTISTS values (0003, 'Charles', 'Darwin', 650000);
SQL> insert INTO SCIENTISTS values (0004, 'Curie', 'Einstein', 550000);
SQL> commit;
```

3. Verify the inserted data.

```
SQL> select * from SCIENTISTS;
```

4. Encrypt the `Salary` column from SCIENTISTS.

```
SQL> alter table SCIENTISTS modify (Salary ENCRYPT);
```

5. Transparent Data Encryption decrypts the encrypted column automatically and returns the data in clear format.

```
SQL> select salary from SCIENTISTS;
```

6. Verify the column is encrypted in your DB.

```
SQL> select * from DBA_ENCRYPTED_COLUMNS;
```

7. View the information of the HSM keystore.

```
SQL> select * from V$ENCRYPTION_WALLET;
```

6.2 Creating an Encrypted Tablespace

1. To create an encrypted tablespace, you must use the CREATE TABLESPACE statement with the ENCRYPTION USING clause.

```
SQL> create tablespace SECURE_TS DATAFILE 'C:/Oraclenew/oradata/ORCLNEW/
SECURETS_01.DBF' SIZE 10M ENCRYPTION USING 'AES256' ENCRYPT;
```

2. Create an EMP table inside the SECURE_TS tablespace.

```
SQL> create table EMP (EMPID NUMBER(4),NAME VARCHAR(100),SALARY NUMBER(6))
tablespace SECURE_TS;
```

3. Add data to the EMP table.

```
SQL> insert into EMP values (0001, 'Michael Jackson', 999999);
SQL> insert into EMP values (0002, 'Lady Gaga', 888888);
SQL> insert into EMP values (0003, 'Freddie Mercury', 777777);
SQL> insert into EMP values (0004, 'Steven Tyler', 666666);
SQL> commit;
```

4. View the data from the EMP table.

```
SQL> select * from EMP;
```

5. Close the keystore.

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

6. Now, try to view the contents of the EMP table.

```
SQL> select * from EMP;
```

As the keystore is closed, you will get an error message "ORA-28365: wallet is not open", and hence you cannot view the data from the EMP table

7. Open the Keystore.

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

8. Now view the data from the EMP table.

```
SQL> select * from EMP;
```

9. List all the Key IDs.

```
SQL> select KEY_ID from V$ENCRYPTION_KEYS;
```

6.3 Encrypting an Existing Tablespace with Online Conversion

To encrypt an existing tablespace with online conversion, use ALTER TABLESPACE with the ONLINE and ENCRYPT clauses.

1. Log in to the DB instance as a system user.

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

2. Open the HSM Keystore.

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

3. Create a tablespace.

```
SQL> create tablespace NONSECURE_TS DATAFILE 'C:/Oraclenew/oradata/ORCLNEW/
NONSECURETS_01.DBF' SIZE 10M;
```

4. Create EMP table inside the NONSECURE_TS tablespace.

```
SQL> create table EMP (EMPID NUMBER(4),NAME VARCHAR(100),SALARY NUMBER(6))
tablespace NONSECURE_TS;
```

5. Add data to the EMP table.

```
SQL> insert into EMP values (0001, 'Michael Jackson', 999999);
SQL> insert into EMP values (0002, 'Lady Gaga', 888888);
SQL> insert into EMP values (0003, 'Freddie Mercury', 777777);
SQL> insert into EMP values (0004, 'Steven Tyler', 666666);
SQL> commit;
```

6. View the data from the EMP table.

```
SQL> select * from EMP;
```

7. Check that the COMPATIBLE parameter is set correctly according to the DB version.

```
SQL> show PARAMETER COMPATIBLE;
Example
SQL> show PARAMETER COMPATIBLE;
NAME TYPE VALUE
------------ ----------- ------
compatible string 23.6.0
```

8. Encrypt the NONSECURE_TS tablespace.

```
SQL> alter tablespace NONSECURE_TS ENCRYPTION ONLINE using 'AES192' ENCRYPT
FILE_NAME_CONVERT = ('NONSECURETS_01.DBF','SECURETS_02.DBF');
```

9. View data from the EMP table.

```
SQL> select * from EMP;
```

10. Verify the NONSECURE_TS tablespace got encrypted.

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

11. Close the keystore.

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

Now, try to view the contents of the EMP table; nothing is displayed as the wallet is closed.

As the keystore is closed, you will get an error message “ORA-28365: wallet is not open” and hence you cannot view the data from the EMP table.

12. Open the keystore.

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

13. Verify the EMP table.

```
SQL> select * from EMP;
```

6.4 Decrypting an Existing Tablespace with Online Conversion

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

1. Log in to the DB instance as a system user.

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

2. List the encrypted tablespace.

```
SQL> select TABLESPACE_NAME, ENCRYPTED from DBA_TABLESPACES;
```

3. Open a Keystore.

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

4. Decrypt the existing tablespace NONSECURE_TS.

```
SQL> alter tablespace NONSECURE_TS ENCRYPTION ONLINE DECRYPT FILE_NAME_CONVERT =
('SECURETS_02.DBF', 'NONSECURETS_02.DBF');
```

5. Verify that the tablespace is no longer encrypted.

```
SQL> select TABLESPACE_NAME, ENCRYPTED from DBA_TABLESPACES WHERE
 TABLESPACE_NAME='NONSECURE_TS';
```

6.5 Rekeying an Existing Tablespace with Online Conversion

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

1. Log in to the DB instance as a system user.

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

2. List the encrypted tablespace.

```
SQL> select TABLESPACE_NAME, ENCRYPTED from DBA_TABLESPACES;
```

3. Open a Keystore.

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

4. Rekey the existing tablespace SECURE_TS.

```
SQL> alter tablespace SECURE_TS ENCRYPTION USING 'AES192' REKEY FILE_NAME_CONVERT
= ('SECURETS_01.DBF','SECURETS_03.DBF');
```

5. Check if the rekeying was successful..

```
SQL> select * from V$ENCRYPTED_TABLESPACES;
```

6.6 Encrypting an Existing Tablespace with Offline Conversion

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

1. Log in to the DB instance as a system user.

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

2. Open HSM Keystore.

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

3. Create a tablespace.

```
SQL> create tablespace NONSECURE01_TS DATAFILE 'C:/Oraclenew/oradata/ORCLNEW/
NONSECURETS_01.DBF' SIZE 10M;
```

4. Create an EMP table inside the NONSECURE01_TS tablespace.

```
SQL> create table EMP (EMPID NUMBER(4),NAME VARCHAR(100),SALARY NUMBER(6))
tablespace NONSECURE01_TS;
```

5. Insert data into the EMP table.

```
SQL> insert into EMP values (0001, 'Michael Jackson', 999999);
SQL> insert into EMP values (0002, 'Lady Gaga', 888888);
SQL> insert into EMP values (0003, 'Freddie Mercury', 777777);
SQL> insert into EMP values (0004, 'Steven Tyler', 666666);
SQL> commit;
```

6. View data from the EMP table.

```
SQL> select * from EMP;
```

7. Bring the NONSECURE01_TS tablespace offline.

```
SQL> alter tablespace NONSECURE01_TS OFFLINE NORMAL;
```

8. Encrypt the NONSECURE01_TS tablespace.

```
SQL> alter tablespace NONSECURE01_TS ENCRYPTION OFFLINE ENCRYPT;
```

Alternatively, use the ALTER DATABASE DATAFILE SQL statement to encrypt individual data files within a tablespace.

```
SQL> alter database DATAFILE ' NONSECURETS01_TS.DBF' ENCRYPT;
```

9. Bring the tablespace online.

```
SQL> alter tablespace NONSECURE01_TS ONLINE;
```

10. Close the keystore.

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

11. Now, try to view the contents of the EMP table; nothing is displayed as the wallet is closed.

```
SQL> select * from EMP;
```

6.7 Decrypting an Existing Tablespace with Offline Conversion

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

1. Log in to the DB instance as a system user.

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

2. List the encrypted tablespace.

```
SQL> select TABLESPACE_NAME, ENCRYPTED from DBA_TABLESPACES;
```

3. Open a Keystore.

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

4. Bring the tablespace NONSECURE01_TS offline.

```
SQL> alter tablespace NONSECURE01_TS OFFLINE NORMAL;
```

5. Decrypt the existing tablespace NONSECURE01_TS.

```
SQL> alter tablespace NONSECURE01_TS ENCRYPTION OFFLINE DECRYPT;
```

6. Bring the tablespace NONSECURE01_TS online.

```
SQL> alter tablespace NONSECURE01_TS ONLINE;
```

7. Verify the tablespace got decrypted.

```
SQL> select TABLESPACE_NAME, ENCRYPTED from DBA_TABLESPACES;
```

6.8 Migrating the Master Encryption Key from Software Keystore to the HSM Keystore

This section describes how the software wallet is migrated to a hardware-based wallet.

We have used Windows Server for this description. The SQL commands use Windows-style paths; change the paths according to the appropriate operating system.

6.8.1 Creating a Software Keystore

This example starts by creating a database protected by a software wallet. If you already have a database protected by a software wallet, you can skip this section.

1. Create a wallet directory in the C:\oracle\admin\ORCLDB\WALLET directory, e.g., wallet.

2. Log in to the database instance as a user granted the SYSDBA administrative privilege.

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

3. Set the WALLET_ROOT parameter.

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

4. Shut down and start up the database.

```
SQL> shutdown immediate;
SQL> startup;
```

5. Set the TDE_CONFIGURATION parameter.

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

6. Grant the ADMINISTER KEY MANAGEMENT or SYSKM privilege to SYSTEM and any user you want.

```
SQL> grant ADMINISTER KEY MANAGEMENT to system;
SQL> commit;
```

7. Connect to the database as a system user.

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

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

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

9. Run the ADMINISTER KEY MANAGEMENT SQL statement to open the software-based keystore.

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

10. Set the Master Encryption Key in the software keystore.

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

11. Create a SCIENTISTS table in the DB.

```
SQL> create table SCIENTISTS (SCID NUMBER(4), FirstName VARCHAR2(128), LastName
VARCHAR2(128), Salary NUMBER(6));
```

12. Add data to the SCIENTISTS table.

```
SQL> insert into SCIENTISTS values (0001, 'Albert', 'Einstein', 850000);
SQL> insert into SCIENTISTS values (0002, 'Isaac', 'Newton', 750000);
SQL> insert into SCIENTISTS values (0003, 'Charles', 'Darwin', 650000);
SQL> insert INTO SCIENTISTS values (0004, 'Curie', 'Einstein', 550000);
SQL> commit;
```

13. Verify the added data in the SCIENTISTS table.

```
SQL> select * from SCIENTISTS;
```

14. Encrypt the 'Salary' column from SCIENTISTS.

```
SQL> alter table SCIENTISTS modify (Salary ENCRYPT);
```

15. The Transparent Data Encryption decrypts the encrypted column automatically and returns the data in clear format.

```
SQL> select salary from SCIENTISTS;
```

16. Verify the column is encrypted in your DB.

```
SQL> select * from DBA_ENCRYPTED_COLUMNS;
```

17. View the information of the software keystore.

```
SQL> select * from V$ENCRYPTION_WALLET;
```

18. Create an encrypted tablespace.

```
SQL> create tablespace SECURETS DATAFILE 'C:/Oraclenew/oradata/ORCLNEW/
SECURETDB_01.DBF' SIZE 10M ENCRYPTION USING 'AES256' ENCRYPT;
```

19. Create the EMP table inside the SECURETS tablespace.

```
SQL> create table EMP (EMPID NUMBER(4),NAME VARCHAR(100),SALARY NUMBER(6))
tablespace SECURETS;
```

20. Add data to the EMP table.

```
SQL> insert into EMP values (0001, 'Michael Jackson', 999999);
SQL> insert into EMP values (0002, 'Lady Gaga', 888888);
SQL> insert into EMP values (0003, 'Freddie Mercury', 777777);
SQL> insert into EMP values (0004, 'Steven Tyler', 666666);
SQL> commit;
```

21. View the data from the EMP table.

```
SQL> select * from EMP;
```

22. Close the software keystore.

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

23. Now, try to view the contents of the EMP table.

```
SQL> select * from EMP;
```

As the keystore is closed, you will get an error message “ORA-28365: wallet is not open” and hence you cannot view the data from the EMP table.

24. Open the Keystore.

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

25. Now, view the data from the EMP table.

```
SQL> select * from EMP;
```

26. List all Key IDs.

```
SQL> select KEY_ID from V$ENCRYPTION_KEYS;
```

6.8.2 Migrating the Software Keystore to the Utimaco HSM

1. Log in to the database as a user granted the SYSDBA administrative privilege.

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

2. Set the WALLET_ROOT parameter.

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

3. Shut down and start up database.

```
SQL> shutdown immediate;
SQL> startup;
```

4. Set the TDE_CONFIGURATION parameter.

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

5. Connect to the database as a system user.

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

6. Now, migrate the wallet to the HSM using the command below.

```
SQL> ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY <hsm_password>
MIGRATE USING <software_keystore_password> WITH BACKUP USING 'backupdb';
```

7. Now, verify that the wallet has been moved to the HSM wallet using the command below.

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

8. The Transparent Data Encryption decrypts the encrypted column automatically and returns the data in clear format.

```
SQL> select salary from SCIENTISTS;
SQL> select salary from EMP;
```

9. Verify that the column is encrypted in your DB.

```
SQL> select * from DBA_ENCRYPTED_COLUMNS;
```

10. View the information of the keystore.

```
SQL> select * from V$ENCRYPTION_WALLET;
```

11. Change the password of the software keystore to be the same as the HSM password.

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

12. Close the keystore.

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

13. Open the keystore.

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

14. View the information of the keystore.

```
SQL> select * from V$ENCRYPTION_WALLET;
```

6.9 Configuring TDE with Pluggable DB using HSM

6.9.1 About Containers in a CDB

In a multitenant container database (CDB), a container represents a collection of schemas, objects, and associated structures. Each container within a CDB is uniquely identified by an ID and a name.

A CDB can include zero, one, or multiple user-created pluggable databases (PDBs) and application containers. A PDB is a self-contained, portable set of schemas, schema objects, and non-schema objects that appears to Oracle Net clients as an independent database.

An application container is an optional component created by users within a CDB. It holds both data and metadata for one or more application back ends. A CDB may contain zero or more application containers.

6.9.2 About PDBs

A Pluggable Database (PDB) is a user-created collection of schemas, objects, and related structures that logically appears to client applications as an independent database.

Regardless of who creates the PDB, it is always owned by the SYS user. SYS is a common user within the Container Database (CDB), meaning it maintains the same identity across the root container and all existing or future PDBs.

PDBs are designed to be plugged into CDBs, allowing for flexible database management. A single CDB can host multiple PDBs, each of which is accessible over the network as a distinct database.

6.9.3 Configuring TDE with PDB using the HSM

While multiple methods are available to create a Pluggable Database (PDB), the recommended approach is to use the Database Configuration Assistant (DBCA) utility. This explanation assumes that the PDBs have already been created.

While performing this integration, we used Windows Server. The SQL commands use Windows-style paths; change the path according to the appropriate operating system. For the purpose of this guide, we are using the PDB "utimacopdb."

1. Edit the tnsnames.ora file to add a new service for the PDB. By default, the tnsnames.ora file in the location set by the TNS_ADMIN environment variable. Ensure you have correctly set the TNS_ADMIN environment variable to point to the correct tnsnames.ora file.

```
UTIMACOPDB =
(DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
  (CONNECT_DATA =
   (SERVER = DEDICATED)
   (SERVICE_NAME = utimacopdb.localdomain)
)
)
```

2. Restart the Listener Service.

```
#lsnrctl stop
#lsnrctl start
#lsnrctl status
```

3. Log in to the database instance as a user granted the SYSDBA administrative privilege.

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

4. Set the WALLET_ROOT parameter.

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

5. Shut down and start up the database.

```
SQL> shutdown immediate;
SQL> startup;
```

6. Set the TDE_CONFIGURATION parameter.

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

7. Open the hardware keystore in the CDB$ROOT container.

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

8. Set the master encryption key in the CDB$ROOT container on the HSM. If the master encryption key has already been generated on the HSM, skip this step.

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

9. Connect as sysdba.

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

10. Open the PDB in read-write mode.

```
SQL> alter pluggable database <PDB_NAME> open read write;
```

11. Set the container to the PDB.

```
SQL> alter session set container = <pdb_name>;
```

12. Grant the following privileges to the PDB Admin.

```
SQL> grant administer key management to <pdb_admin>;
SQL> grant create session to <pdb_admin>;
SQL> grant connect to <pdb_admin>;
SQL> grant dba to <pdb_admin>;
SQL> grant create any table to <pdb_admin>;
SQL> grant unlimited tablespace to <pdb_admin>;
SQL> alter user <pdb_admin> profile default;
SQL> commit;
```

13. Connect to the PDB using the PDB username.

```
SQL> Connect <pdb_admin>/<system_password>@<Pluggable Database Name>
```

14. Run the ADMINISTER KEY MANAGEMENT SQL statement to open the PDB database.

```
SQL> administer key management set keystore open identified by "<hsm_password>";
```

15. Create the PDB Master Key on the HSM.

```
SQL> administer key management set key identified by "<hsm_password>";
```

6.9.4 Verifying the Master Encryption Key is Encrypting the PDB

1. Create a SCIENTISTS table in the PDB.

```
SQL> create table SCIENTISTS (SCID NUMBER(5), Name VARCHAR(42), SALARY
NUMBER(10));
```

2. Add values to the SCIENTISTS table.

```
SQL> insert into SCIENTISTS values (001, 'George Bailey', 10000);
SQL> insert into SCIENTISTS values (002, 'Denial Vettory', 20000);
SQL> commit;
```

3. Encrypt the Salary column of the SCIENTISTS table.

```
SQL> alter table SCIENTISTS modify (Salary Encrypt);
```

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

```
SQL> select salary from SCIENTISTS;
```

5. List encrypted columns in your databases.

```
SQL> select * from DBA_ENCRYPTED_COLUMNS;
```

6. Create an encrypted tablespace.

```
SQL> create tablespace SECURETS DATAFILE 'C:/Oraclenew/oradata/ORCLNEW/
SECURETS01.DBF' SIZE 10M ENCRYPTION DEFAULT STORAGE (ENCRYPT);
```

7. Create an EMP table inside the NONSECURE_TS tablespace.

```
SQL> create table EMP (EMPID NUMBER(4),NAME VARCHAR(100),SALARY NUMBER(6))
tablespace SECURETS;
```

8. Add data to the EMP table.

```
SQL> insert into EMP values (0001, 'Michael Jackson', 999999);
SQL> insert into EMP values (0002, 'Lady Gaga', 888888); SQL> insert into EMP
values (0003, 'Freddie Mercury', 777777);
SQL> insert into EMP values (0004, 'Steven Tyler', 666666);
SQL> commit;
```

9. View data from the EMP table.

```
SQL> select * from EMP;
```

10. Close the keystore.

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

11. Now, try to view the contents of the EMP table.

```
SQL> select * from EMP;
```

As the keystore is closed, you will get an error message “ORA-28365: wallet is not open” and hence you cannot view the data from the EMP table.

12. Open the Keystore.

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

13. Now, view the data from the EMP table.

```
SQL> select * from EMP;
```

14. List all Key IDs.

```
SQL> select KEY_ID from V$ENCRYPTION_KEYS;
```

6.10 Logs and Validation Steps

Effective logging is essential for monitoring, troubleshooting, and validating system behavior during configuration and testing. Whether you're integrating security modules like PKCS#11 or managing Oracle Database operations, logs provide critical insights into system events, errors, and performance.

This section outlines the key logging mechanisms involved in both the PKCS#11 module and the Oracle Database 23ai environment on Windows Server. It includes guidance on enabling, locating, and interpreting log files to support smooth testing and reliable diagnostics.

6.10.1 PKCS#11 Logs

Enabling PKCS#11 logging to facilitate easier testing and troubleshooting is recommended. This can be done by configuring the Logging Loglevel and LogPath parameters in the configuration file.

- LogPath should point to a writable directory (not a specific file) where log files can be stored.
- Logging Loglevel controls the verbosity of the logs:
    - Set it to 1 for basic logging.
    - For detailed testing and debugging, increase the level to 4.

The generated log file will be named cs_pkcs11_R3.log and located in the directory specified by LogPath. Reviewing this log file can help identify and resolve issues that arise during testing.

Once testing is complete, it is advisable to reduce the Logging Loglevel to 1 or 2 to limit logging to only critical or important messages, thereby optimizing performance and reducing unnecessary log data.

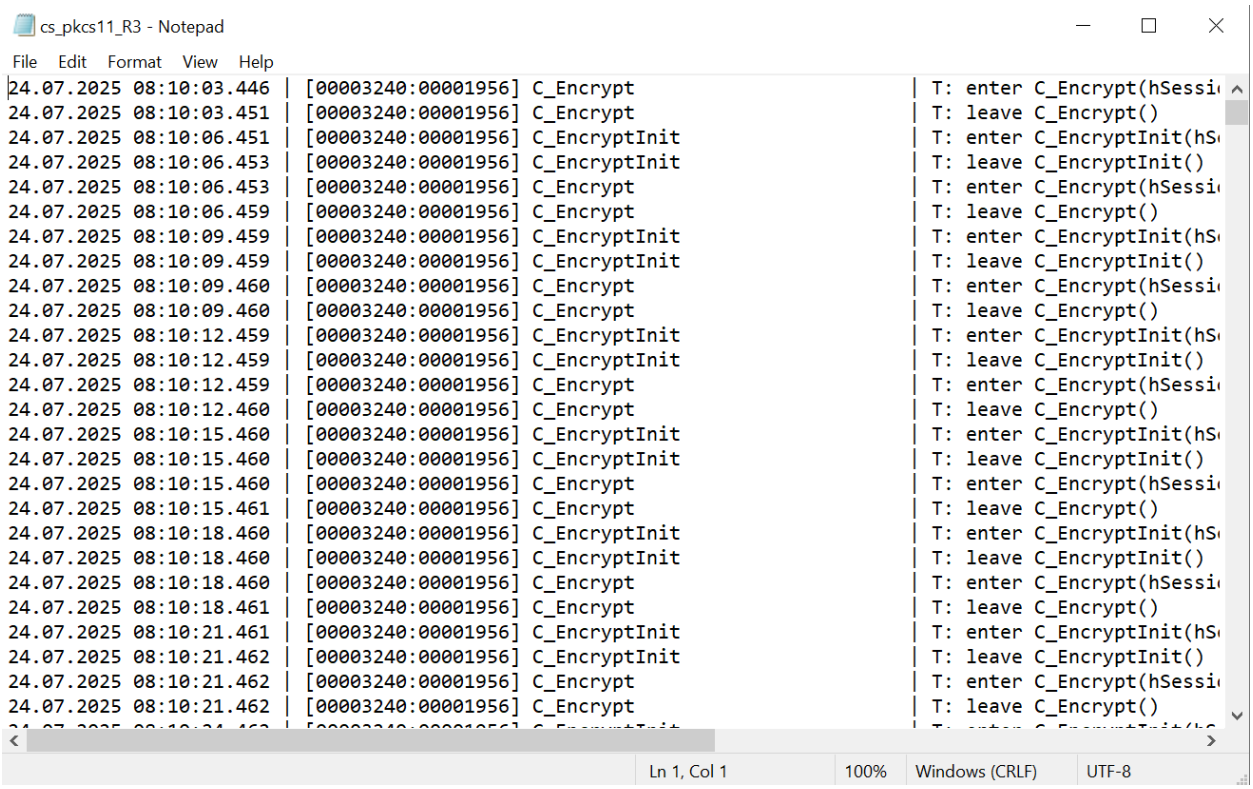

Figure 2 : Sample PKCS#11 Log

6.10.2 Oracle Database Logs

An Oracle Database 23ai on a Windows Server provides several logs and trace files to monitor and troubleshoot database operations. One of the most important among them is the Alert Log.

- Location:

`C:\app\Administrator\product\23ai\diag\rdbms\<SID>\<SID>\trace\alert_<SID>.log`

- Purpose:
  This log records significant database events such as instance startup and shutdown, internal errors, and administrative actions. It is a primary resource for diagnosing issues and ensuring the database works as expected.

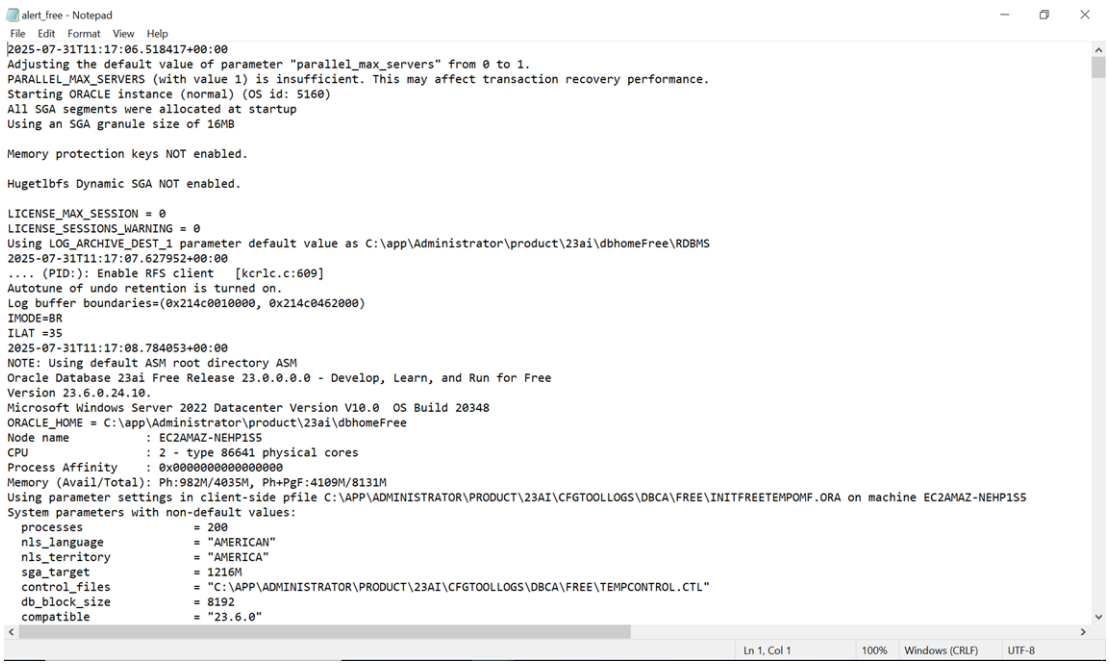

Figure 3 : Sample Oracle Alert Log

7 Optional Features

7.1 Configuring Auto Login for the Hardware Keystore

The Auto Login feature for Oracle wallets allows password-free access, enabling PKI-based authentication to services without human intervention. When enabled, it generates an obfuscated copy of the wallet, which is automatically used until the feature is unavailable. By default, auto login is turned off. You must manually activate auto login to enable single sign-on (SSO) access across multiple Oracle databases. Once enabled, an `.sso` file is created in the wallet directory to facilitate seamless access.

1. Close the Hardware Keystore if it is opened.

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

2. Set the WALLET_ROOT parameter.

```
SQL> connect / as sysdba
SQL> alter system set wallet_root='<path to the oracle wallet directory>'
 scope=spfile
```

3. Shut down and start up the database.

```
SQL> shutdown immediate;
SQL> startup;
```

4. Set the TDE_CONFIGURATION parameter.

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

5. Create the Software Keystore.

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

6. Open the Software Keystore.

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

7. Add the HSM password as a client to the Software Keystore.

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

8. Close the Software Keystore.

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

9. Create the Auto Login keystore.

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

10. Set the TDE_CONFIGURATION parameter.

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

11. At this stage, close the database and open it one more time. The next time a TDE operation executes, the hardware security module Auto Login keystore will open automatically.

```
SQL> shutdown immediate;
SQL> startup;
```

12. Check the status of the wallet.

```
SQL> select * from V$ENCRYPTION_WALLET;
```

Now, you have a software wallet that contains the HSM password, which is protected by Oracle's Auto Login feature.

8 Troubleshooting

8.1 Common Issues and How to Resolve Them

Error	Diagnosis
ORA-46661: keystore not open in root container	You must open it in the CDB root before accessing it from PDBs.
ORA-46658: keystore not open in the container	The keystore is not open in the current container. Ensure it is opened in both CDB and PDB contexts.
ORA-28414: Current master key is in the external keystore (OKV/KMS).	You need proper configuration and access to the external key manager.
ORA-01031: insufficient privileges	You may need `SYSKM`, `SYSDBA`, or `ADMINISTER KEY MANAGEMENT` privileges.
ORA-28365: Wallet is not open.	Use ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN to open it.
SP2-0640: Not connected	SQL*Plus lost connection or was never connected.
ORA-28353: Failed to open wallet.	Could be due to an incorrect path, permissions, or a corrupted wallet.
ORA-01017: invalid credential or not authorized; login denied	Check username/password or authentication method.
ORA-28374: Typed master key not found in wallet	Possibly due to a missing or incorrectly configured Master Key.

ORA-65040: Operation is not allowed from within a pluggable database	Some actions must be taken at the CDB root.
ORA-03113: end-of-file on communication channel	Indicates abrupt disconnection—could be due to network issues, server crash, or client timeout.
ORA-01516: non-existent log file, data file, or temporary file	Check file paths and existence.
ORA-00955: name is already used by an existing object	Rename or drop the existing object before creating a new one.
ORA-28426: must FINISH a tablespace encrypt, decrypt or rekey command first	Use ALTER TABLESPACE ... ENCRYPT FINISH before starting a new one.

Table 9: List of Errors and their Diagnosis

8.2 Log Locations and Interpretation

Understanding where logs are stored is essential for effective troubleshooting. Below are the key log file locations for PKCS#11 integration and Oracle Database 23ai on Windows Server.

8.2.1 PKCS#11 Log File

- Log File Name: cs_pkcs11_R3.log
- Location: Defined by the LogPath parameter in the PKCS#11 configuration file.

Example: C:\ProgramData\Utimaco\PKCS11_R3\

- Details:
  This log captures detailed information about PKCS#11 operations, including initialization, cryptographic actions, and error messages. The Logging Loglevel setting controls the verbosity.

8.2.2 Oracle Database Alert Log

- Log File Name: alert_<SID>.log
- Location:
  C:\app\Administrator\product\23ai\diag\rdbms\<SID>\<SID>\trace\
- Details:
  The alert log records major database events such as instance startup/shutdown, internal errors, and administrative actions. It is a primary resource for diagnosing database-related issues.

9 Contact and Support Information

You can reach us from Monday to Friday, 09.00 a.m. to 05.00 p.m., Central European Time (CET).

Utimaco IS GmbH
Germanusstr. 4
52080 Aachen
Germany

RMA Query

If you need to send the device back to Utimaco IS GmbH, please open a new RMA case (Return Merchandise Authorization). We request that you use the following web address. RMA cases cannot be opened by email or phone.

https://support.hsm.utimaco.com/support/rma/new

Other Support Queries

- Mail (preferred contact method)
  support@utimaco.com
  Attach the diagnostic information to your email.
- Web portal
  https://support.hsm.utimaco.com/support/cases/new/
  The diagnostic information will be requested in our response if necessary.
- By phone
  AMERICAS +1-844-UTIMACO (+1 844-884-6226)
  EMEA +49 800-627-3081
  APAC +81 800-919-1301
  The diagnostic information will be requested in our response if necessary.

10 Appendices

10.1 References (links to external docs)

Title	Document Number
utrust_Anchor_LAN_V5_Operating_Manual	2021-0039
utrust_Anchor_PCIe_Operating_Manual	2020-0042
CryptoServerLAN_V5_Operating_Manual	2018-0004
CryptoServerPCIe_CSe-Series_Operating_Manual	M013-0002-en
CryptoServerPCIe_Se-Series_Gen2_Operating_Manual	M015-0001-en

Table 10: References