Oracle

DB 19c on Oracle Linux

Integration Guide

CryptoServer

4.30.0

utimaco®

Imprint

Copyright 2020	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	06/10/2025
Status	PUBLISHED
Document No.	IG-2025-0021

Table of Contents

1 Introduction

Thank you for purchasing our CryptoServer security system. We hope you are satisfied with our product. This paper provides an integration guide explaining how to configure an Utimaco CryptoServer Hardware Security Module (HSM) with Oracle DB 19c on Oracle Linux 7.6.

Please do not hesitate to contact us if you have any complaints or comments.

1.1 About this Guide

This guide describes how to enable HSM integration with Oracle database server including Oracle database installation.

1.1.1 Target Audience for this Guide

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

1.1.2 Document Conventions

We use the following conventions in this guide:

Convention	*Use*	*Example*
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press the OK button.
`Monospaced`	File names, folder and directory names, commands, file outputs, programming code samples	You will find the file example.conf in the /exmp/demo/ directory.
Italic	References and important terms	See Chapter 3, "Sample Chapter", in the *CryptoServer - csadm Manual* or [CSADMIN].

Table 1: Document conventions

Special icons are used to highlight the most important notes and information.

Here you find important safety information that should be followed.

Here you find additional notes or supplementary information.

1.1.3 Abbreviations

We use the following abbreviations in this guide:

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

Abbreviation	*Meaning*
JRE	Java Runtime Environment
MBK	Master backup key
P11CAT	the PKCS#11 graphical interface tool
CXI	Cryptographic eXtended Interface
FIPS	Federal Information Processing Standards

Table 2: List of Abbreviations

2 Overview

The Oracle DB is one of the most trusted and widely-used relational database engines. The system is built around a relational database framework in which data objects may be directly accessed by users through a structured query language. Oracle is fully scalable and is often used by global enterprises, which manage and process data across wide and local area networks. The Oracle database has its own network component to allow communications across networks.

Oracle provides a straightforward method of managing database credentials across multiple domains by using Oracle Wallets. These enable users to update the database credentials, without the need to change specific data source definitions, since the database connection string in the data source definition is resolved by an entry in the wallet.

If the security of the wallets and cryptographic material they contain needs to be enhanced, the Oracle Database needs to be configured to use a Hardware Security Module (HSM). When the HSM module is enabled with the Oracle Database, this strengthens the protection of the wallets.

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

3 Prerequisites and Requirements

Ensure that the system environment you will be using, meets the following hardware and software requirements.

3.1 Software Requirements

Software	*Software Requirements*
Operating system	CentOS 7,8 Red Hat Enterprise Linux: 7,8
Oracle 19c	Recommended version of Oracle
Oracle Linux	Oracle Linux 7.x
JRE 8	Java Runtime Environment 8

Table 3: List of Software Requirements

3.2 Hardware Requirements

Hardware	*Hardware Requirements*
Utimaco LAN HSM	CryptoServer CSe-Series/Se-Series LAN with firmware SecurityServer 4.30.0
Utimaco PCI-e HSM	CryptoServer CSe-Series/Se-Series PCI-e with firmware SecurityServer 4.30.0

Table 4: List of Hardware Requirements

4 Installing Oracle Database

Description of the installation of the Oracle database software is included in this guide for informational purposes only.

Installation is done by using https://oracle-base.com/articles/12c/oracle-db-12cr1-installation-on-oracle-linux-7

Note that Oracle 19c on Oracle Linux 7.x. presents an unstable environment. Oracle database might lose connection or stop working at certain times. These problems are not related to the HSM connection.

4.1 Preparing the Server for Oracle Installation

4.1.1 Preparing Hosts File

1. The /etc/hosts file must contain a fully qualified name of the server. You can check your fully qualified name by typing the command:

```
›_ Console
# hostname
```

1. Change the hosts file:

```
›_ Console
# nano /etc/hosts
```

1. Make sure that the document has a line in the following format:

<IP-address> <machine-name> <fully-qualified-machine-name>

For example:

192.168.2.2 oraclelinux19c oraclelinux19c.localdomain

1. You can find the machine-name by checking at what follows after the @ character on the terminal. For example:

›_ Console

#root@oraclelinux19c

1. Set or change the correct hostname (fully qualified name) in the /etc/hostname file by typing the command:

›_ Console
nano /etc/hostname

1. Make sure that the document has the following line (change "oraclelinux19c" with your machine name):

oraclelinux19c.localdomain

1. You can find your IP address by using the ifconfig command.

›_ Console
ifconfig

4.1.2 Installing Oracle 19c Prerequisites

›_ Console

```
# yum install -y oracle-database-preinstall-19c
```

Optionally perform a full update:

›_ Console

```
# yum update -y
```

4.1.3 Additional Setup

1. Set the password for the "oracle" user.

›_ Console

```
# passwd oracle
```

1. Set secure Linux value to "permissive" by editing the `/etc/selinux/config` file and making sure that the `SELINUX` flag is set as follows.

›_ Console

```
# nano /etc/selinux/config

# SELINUX=permissive
```

1. Once the change is complete, restart the server or run the following command.

›_ Console

```
# setenforce Permissive
```

1. If you have the Linux firewall enabled, you will need to disable it.

```
›_ Console
# systemctl stop firewalld
# systemctl disable firewalld
```

4.1.4 Creating Directories

Create the directories in which the Oracle software will be installed.

```
›_ Console
# mkdir -p /u01/app/oracle/product/19.0.0/dbhome_1
# mkdir -p /u02/oradata
# chown -R oracle:oinstall /u01 /u02
# chmod -R 775 /u01 /u02
```

4.1.5 Bash

1. Create a $_{scripts}$ directory.

```
›_ Console
# mkdir /home/oracle/scripts
```

1. Create an environment file called `setEnv.sh`. The "$" characters are avoided by using "\". If you don't create the file with the `cat` command, you will need to remove the escape characters. Make sure to change the hostname accordingly.

›_ Console

```
# Oracle Settings cat > /home/oracle/scripts/setEnv.sh <<EOF

# Oracle Settings export TMP=/tmp export TMPDIR=\$TMP

export ORACLE_HOSTNAME=oraclelinux.localdomain export ORACLE_UNQNAME=orcl export ORACLE_BASE=/u01/app/oracle export ORACLE_HOME=\$ORACLE_BASE/product/19.0.0/dbhome_1 export ORA_INVENTORY=/u01/app/oraInventory export ORACLE_SID=orcl export PDB_NAME=pdb1 export DATA_DIR=/u02/oradata

export PATH=/usr/sbin:/usr/local/bin:\$PATH export PATH=\$ORACLE_HOME/bin:\$PATH

export LD_LIBRARY_PATH=\$ORACLE_HOME/lib:/lib:/usr/lib export CLASSPATH=\$ORACLE_HOME/jlib:\$ORACLE_HOME/rdbms/jlib

EOF
```

1. Add the following lines at the end of the `/home/oracle/.bash_profile` file.

›_ Console

```
# Oracle set environment

. /home/oracle/scripts/setEnv.sh
```

1. Create a `start_all.sh` and `stop_all.sh` script that can be called from a startup or shutdown service. Make sure the ownership and permissions are correct.

›_ Console
chown -R oracle:oinstall /home/oracle/scripts

```
# chmod u+x /home/oracle/scripts/*.sh

cat > /home/oracle/scripts/start_all.sh <<EOF
#!/bin/bash
. /home/oracle/scripts/setEnv.sh

export ORAENV_ASK=NO
. oraenv
export ORAENV_ASK=YES

dbstart \$ORACLE_HOME
EOF

cat > /home/oracle/scripts/stop_all.sh <<EOF
#!/bin/bash
. /home/oracle/scripts/setEnv.sh

export ORAENV_ASK=NO
. oraenv
export ORAENV_ASK=YES

dbshut \$ORACLE_HOME
EOF
```

1. Create a Linux service to automatically start or stop the database. Create the service file called dbora.service.

›_ Console

```
#nano /lib/systemd/system/dbora.service
```

1. Add the following lines:

›_ Console

```
[Unit]
Description=The Oracle Database Service
After=syslog.target network.target
[Service]
# systemd ignores PAM limits, so set any necessary limits in the service.
# Not really a bug, but a feature.
# https://bugzilla.redhat.com/show_bug.cgi?id=754285
LimitMEMLOCK=infinity
LimitNOFILE=65535

#Type=simple
# idle: similar to simple, the actual execution of the service binary is delayed
#until all jobs are finished, which avoids mixing the status output with shell output of services.
RemainAfterExit=yes
User=oracle
```

Group=oinstall

Restart=no

ExecStart=/bin/bash -c '/home/oracle/scripts/start_all.sh'

ExecStop=/bin/bash -c '/home/oracle/scripts/stop_all.sh'

[Install]

WantedBy=multi-user.target

1. Reload the system so it can see the new service.

›_ Console
systemctl daemon-reload

1. Start the service and enable it for an automatic restart on reboot.

›_ Console
systemctl start dbora.service # systemctl enable dbora.service # systemctl status dbora.service

1. Reboot the machine.

4.2 Installing the Database

4.2.1 Preparing the Installation Files

1. Login as an Oracle user.
2. Copy **LINUX.X64_193000_db_home.zip** to **/tmp** folder.

›_ Console

```
# cd $ORACLE_HOME
```

3. Install unzip (if necessary) and unzip the **LINUX.X64_193000_db_home.zip** (which you can find on the Oracle website. For example: https://oracle-base.com/articles/19c/oracle-db19c-installation-on-oracle-linux-7) to **$ORACLE_HOME**.

›_ Console

```
# sudo yum install unzip
```

```
# unzip -oq /tmp/LINUX.X64_193000_db_home.zip
```

4. Start the installation.

›_ Console

```
#./runInstaller
```

4.2.2 Continuing the Installation

1. Select **Create and configure a single instance database**, as shown in Figure 1. Click on **Next**.

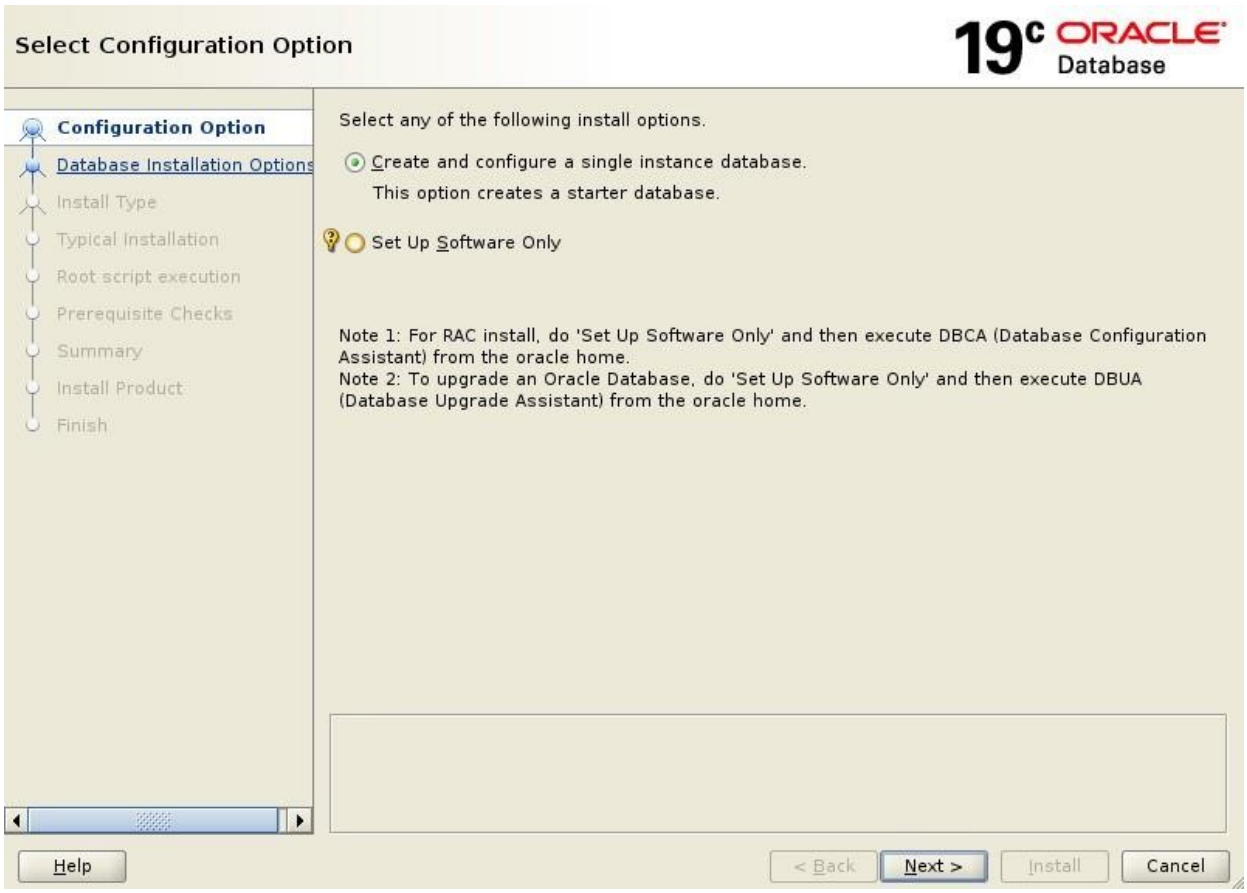

Figure 1: Install Oracle database

2. Choose between **Desktop class** and **Server class** options. Click on **Next**.
3. Set the basic configuration of your new Oracle DB installation. Carefully check that all settings are correct and click on **Next**.

Be sure to select Enterprise Edition that supports TDE!

The Figure 2 shows an example of such a configuration. These settings affect the PKCS#11 installation and are provided here to help you understand the scripts used later in the configuration of the server and PKCS#11.

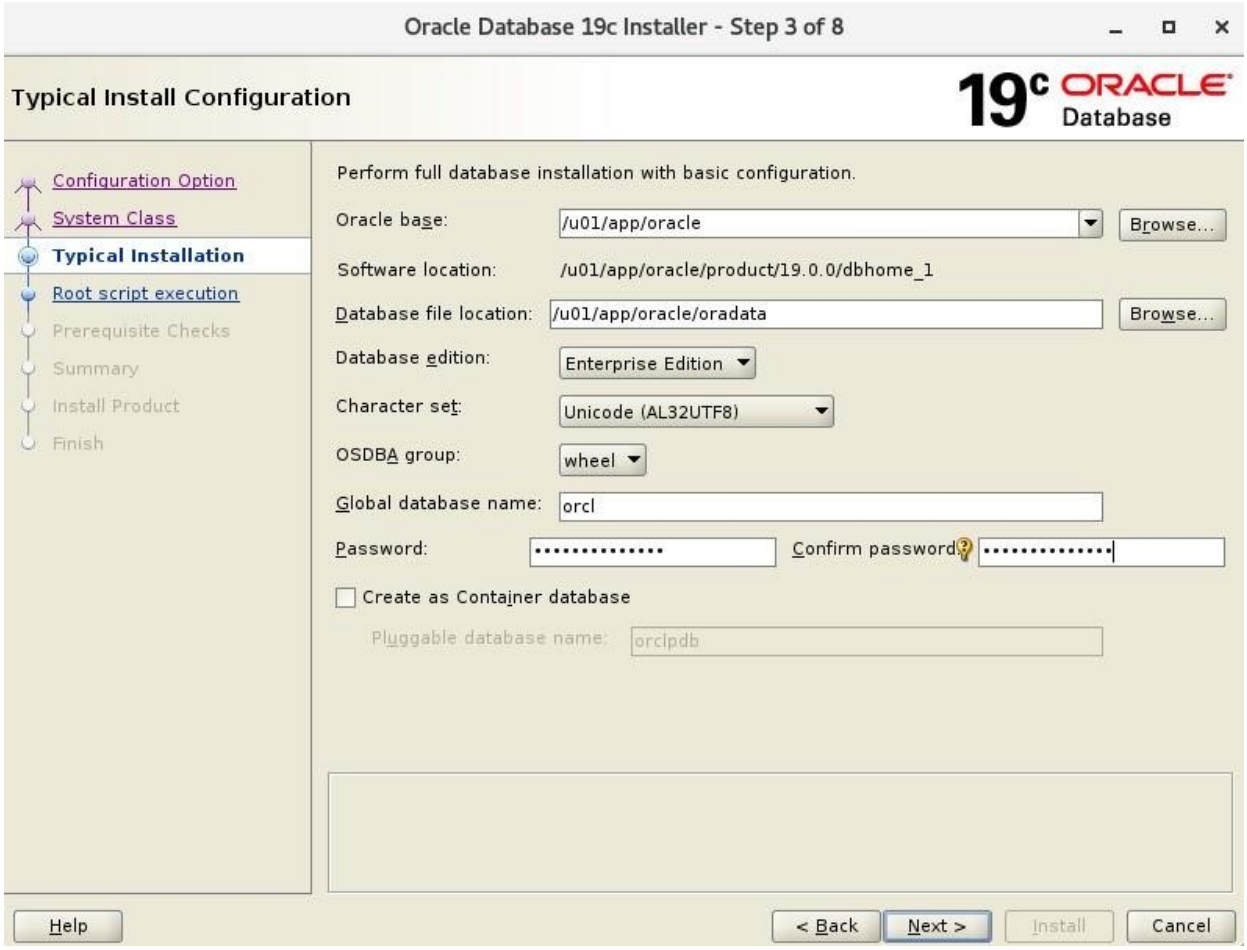

Figure 2: Install Oracle database

4. Set the directory for the installation of metadata files and the operating system group, as shown in Figure 3. Click on **Next**.

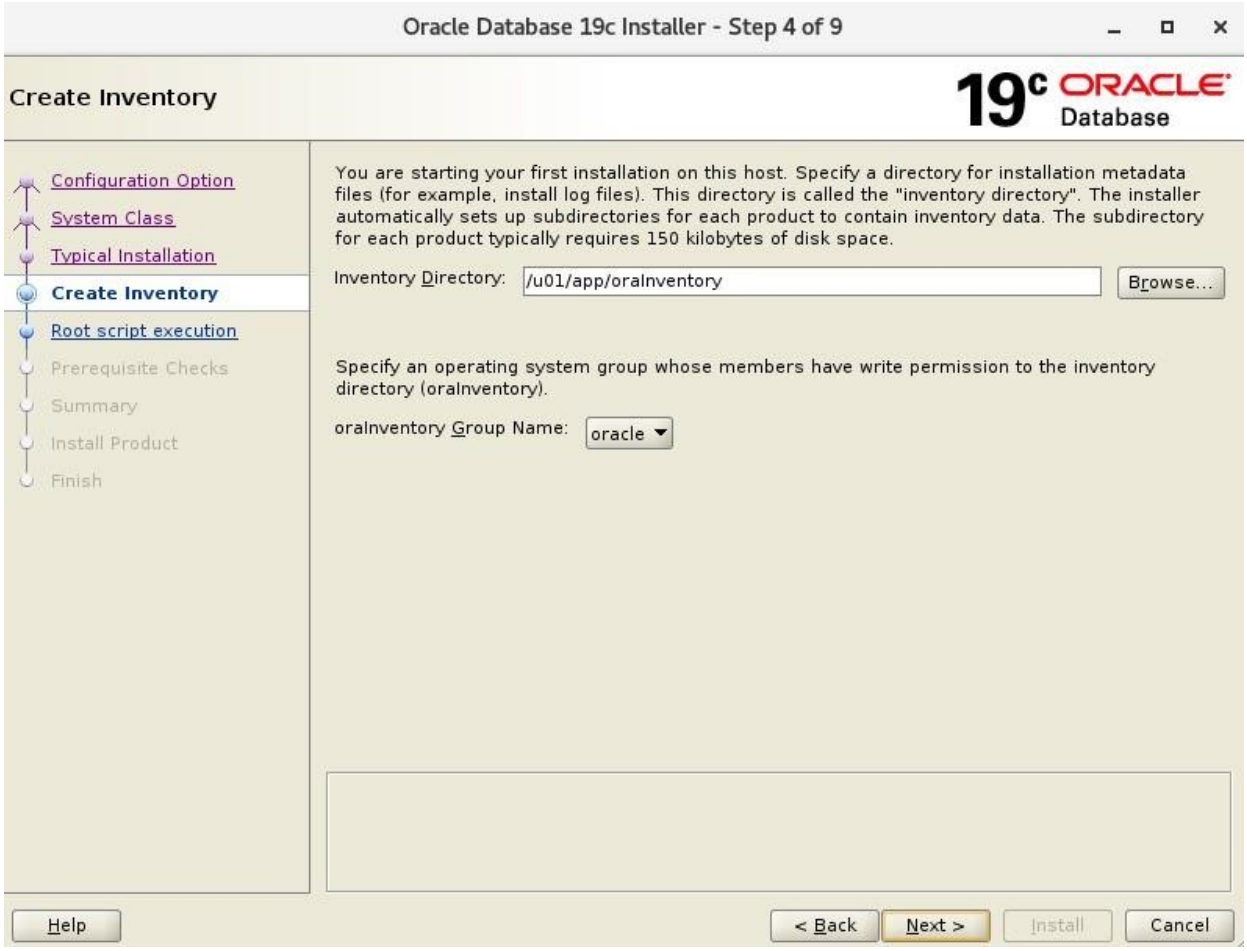

Figure 3: Install Oracle database

5. Provide root user credentials for the script execution, as shown in Figure 4. Click on **Next**.

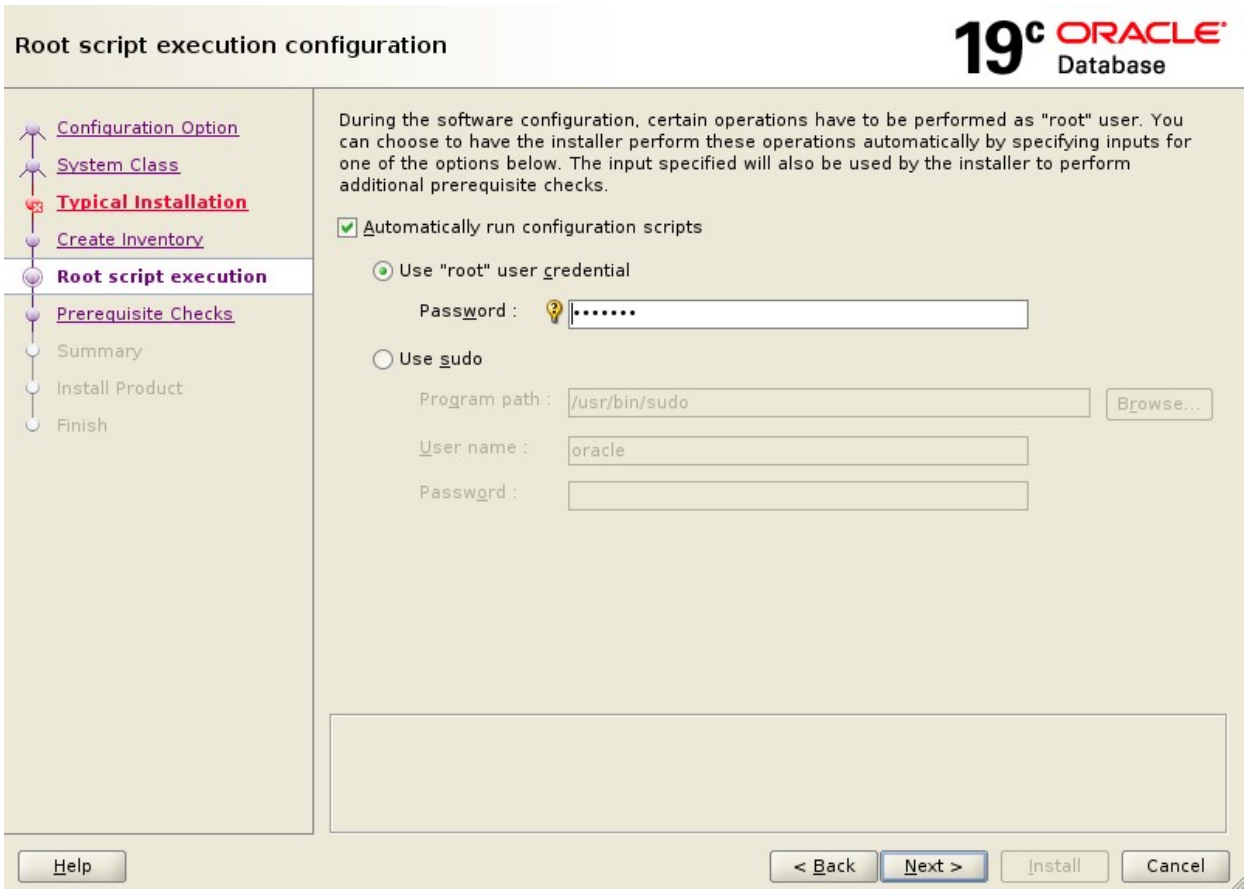

Figure 4: Install Oracle database

1. Prerequisite checks are performed.
2. Check the summary of all installation parameters. When satisfied, click on **Install**.
3. Wait for the installation to finish.
4. Click on **Close**.

5 Configuring PKCS#11

5.1 Introduction and Prerequisites

Before the PKCS#11 interface and library can be used, some manual actions have to be performed. Follow the steps below to configure the PKCS#11 library and initialize a PKCS#11 slot. For further information about the PKCS#11 setup, please refer to [CSPKCSDG].

Oracle DB will load only one PKCS#11 library!

5.2 Installing PKCS#11 on the Workstation

The installation file for the PKCS#11 is located on the Product CD. The installer creates an environment variable called CS_PKCS11_R2_CFG. It contains the path to Utimaco's PKCS#11 configuration file. By default, we have to copy the file to /etc/utimaco in Linux. We have to set the CS_PKCS11_R2_CFG environment variable to point to this location.

In order to be able to access the HSM via PKCS#11, the configuration file needs to be modified.

1. Set the path to the logfile and set the desired log level.

```
[Global]

# For unix:
Logpath = /tmp

# For windows:
# Logpath = C:/ProgramData/Utimaco/PKCS11_R2

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

2. Set the IP address of the HSM.

```
[CryptoServer]

# Device specifier (here: CryptoServer is CSLAN with IP address 127.0.0.1)
```

```
Device = 127.0.0.1
```

3. Optionally, make additional modifications to the configuration file, such as setting up an external store as described in [CSPKCSM]. We suggest to modify the PKCS#11 config file to KeepAlive flag active.

```
[Global]

# Prevents expiring session after inactivity of 15 minutes

KeepAlive = true
```

5.3 Generating an MBK

One of the steps of the HSM initialization is generating a new MBK, which can be used for creating backups, for using an external storage and for synchronizing HSM clusters. By default, MBK is an AES256 key, though it is also possible to generate a DES MBK by using the csadm tool for backward compatibility reasons.

It is required to generate an MBK for the HSM to become operational. Without an MBK no cryptographic operations can be run.

To generate an MBK:

1. Open the Crypto Administration Tool.
2. Achieve the permission level of at least 02000000.
3. Click **Manage MBK** to access the **Remote Master Backup Key Management window** and select the Generate tab.
4. Type the name of the MBK in the **MBK Name** section.
5. Select the backup mode of the MBK shares as either **XOR** or **m out of n**.
    a. If **m out of n** was selected it is necessary to select the number of m (shares) and n (shares) by using the drop-down menus, set by default as 2 and 3.
6. In case that this MBK also needs to be imported at the same time into the HSM, select the **Automatic MBK Import** option.

7. Click Generate.
8. Select whether the MBK shares should be saved on smartcards or as keyfiles by selecting either the **Smartcard Token** or the **Keyfile Token** option.
    a. If you chose to export the MBK shares on smartcards, follow the instructions on the smart card reader to export all of the m parts.

5.4 Importing the MBK

In case the MBK was not imported, when it was generated, or if you want to upload it to another HSM, you need to import it from the keyfiles or smartcards that carry its parts. The MBK was divided into multiple parts by using Shamir's Secret Sharing or XOR and can be restored by using the "m out of n" or XOR principle.

1. Make sure that at least m out of n smart cards/keyfiles are available.
2. Open the CryptoServer Administration Tool.
3. Achieve the permission level of at least 02000000.
4. Click **Manage MBK** to access the **Remote Master Backup Key Management window** and select the Import tab.
5. Select the type of MBK that will be imported and the value m.
6. Click **Import**.
7. Select whether to save the MBK parts on smartcards or as keyfiles by selecting either S**martcard Token** or **Keyfile Token** option.
8. Follow the instructions to import all of the m parts.

5.5 Initializing PKCS#11 on the HSM

In addition to PKCS#11, the PKCS#11 graphical interface tool (P11CAT) and the PKCS#11 command line interface (p11tool2) are installed as well. This chapter shows how to use the P11CAT in order to initialize the PKCS#11 Slot 0. There are 10 active PKCS#11 slots by default. The number of PKCS#11 slots can be modified in the PKCS#11 configuration file.

1. Make sure that the PKCS#11 configuration file contains the IP address of your HSM and that the HSM is running.
2. Open the P11CAT tool on your workstation. When opening the tool for the first time, the slots should be as shown in the Figure 3.

Slot List

Slot ID	Token Init.	PIN Init.	Login Status
0000 0000			
0000 0001			
0000 0002			
0000 0003			
0000 0004			
0000 0005			
0000 0006			
0000 0007			
0000 0008			
0000 0009			

Figure 5: PKCS#11 slots before initialization

3. Select the row **0000 0000** under the Slot ID on the top left-hand side in the Slot List.
4. Click on **Login/Logout**.
5. Click on **Login Generic**.
6. Login as a user with the permission mask at least 20000000 (User Manager permission).
7. Click on **Slot Management**.
8. Create a Security Officer (SO) for Slot 0. Click on **Init Token**. Write the Token Label. Set the SO PIN. Confirm the SO PIN. Click on **Init Token**. Observe the changed Token Init. status for the Slot 0.
9. Logout the ADMIN user. Click on **Login/Logout**. Click on **Logout All**.
10. Login as the SO. Click on **Login/Logout**. Click on **Login SO**. Enter the SO PIN. Click on **Login**.
11. Click on **Slot Management**.
12. Create the User for the Slot 0. Select **Init PIN**. Enter the **Normal User PIN**. Confirm the **Normal User PIN**. Click on Init PIN. Observe the changed PIN Init. status for the Slot 0.

13. Logout the SO. Click on **Login/Logout**. Click on **Logout All**.

The Slot 0 is now initialized. An application or a user can now connect to the PKCS#11 Slot 0 and create or store objects on the slot. Find further information on creating or deleting objects and users in [CSPKCSM] and [LPKCSHD].

5.6 Copy PKCS#11 Library to Oracle DB Library Directory

PKCS#11 library should be stored in the **/opt/oracle/extapi/[32,64]/hsm/{VENDOR}/{VERSION}** directory. The file name should be **libApiName.so**.

We use an operating system with a 64-bit architecture and with the PKCS#11 library **libcs_pkcs11_R2.so**. The directory path is **/opt/oracle/extapi/64/hsm/utimaco/4.30.0**:

›_ Console

```
# mkdir -p /opt/oracle/extapi/64/hsm/utimaco/4.30.0

# chown -R oracle:oinstall /opt/oracle/

# chmod -R 775 /opt/oracle/

# cp libcs_pkcs11_R2.so /opt/oracle/extapi/64/hsm/utimaco/4.30.0/
libcs_pkcs11_R2.so
```

6 Configuring Oracle DB for HSM Wallets

The prerequisites have been prepared: the database server is installed, PKCS#11 is installed and configured. Now the Oracle DB for the use of HSM wallets can be configured.

6.1 Configuring an HSM Wallet

6.1.1 Creating a New Wallet

If no software wallets exist on the system, a new wallet can be created directly on the HSM by using the following commands.

›_ Console

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "1234|
LinuxOracle19c";
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "1234|LinuxOracle19c";
System altered.
```

6.1.2 Start Using an Existing Wallet

1. Use the following command to create the master encryption key:

›_ Console

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "pin[|<token_label>]";

ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "pin[|<token_label>]" [MIGRATE
USING sw_wallet_password]
```

2. You can also use an alternative command.

›_ Console

```
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "pin[|<token_label>]" [MIGRATE USING sw_wallet_password]
```

Where:

- **token_label** is the Token Label used, when initializing the PKCS Token on the HSM. This is optional if the omitted slot 0 will be used.
- **pin** is the pin created for the designated Token.
- **wallet_password** is the password required to open an existing Oracle wallet in the file system. This is used when upgrading the Oracle Wallet software.

Make sure to enclose the **wallet_password** string in double quotation marks (" ").

6.1.3 Check the Newly Created Objects

Check the PKCS#11 log files to see the result of the wallet creation. This log file will also be useful if the wallet creation was not successful. If the log file is empty, the PKCS#11 library was not loaded by the DB server.

Use the PKCS#11 Administration tool or p11tool2 to test the HSM PKCS#11 configuration.

Use the PKCS#11 Administration tool to check the objects, which were stored by the Oracle DB on the HSM during the Wallet initialization

Log into the PKCS tool as a normal user and check the result in the Object Management.

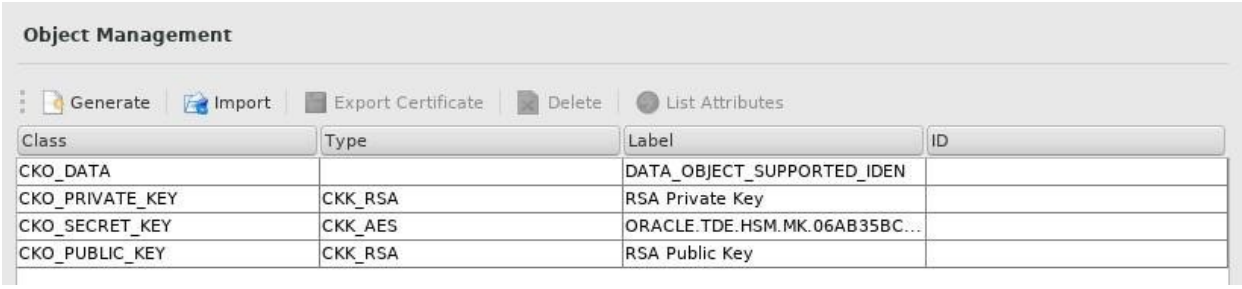

Figure 6: PKCS#11 Objects Created by the Oracle Database

6.2 Specify Wallet Location in Oracle DB

1. The **Sqlnet.ora** file is located in the **$ORACLE_HOME/network/admin** directory.

›_ Console

```
# nano Sqlnet.ora
```

2. Add the following lines:

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

3. Restart the DB server for the changes to take effect.

›_ Console

```
#  sqlplus sys as sysdba
SQL> shutdown immediate
SQL> startup
```

6.3 Migrating Software Oracle Wallet to an HSM Wallet

In order to migrate the software Oracle Wallet, an existing wallet is needed. To create a test wallet, the sqlnet.ora file located in the $ORACLE_HOME/network/admin folder needs to be changed. The following lines need to be added:

```
ENCRYPTION_WALLET_LOCATION =
(SOURCE =
  (METHOD = FILE)
  (METHOD_DATA =
    (DIRECTORY = /u01/app/oracle/admin/orcl/hsmwallet)
  )
)
```

1. When the wallet is created, execute the following commands:

›_ Console

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

SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY password;
```

2. The sqlnet.ora file has to be changed (change METHOD=FILE to METHOD=HSM) and the server restarted.

```
ENCRYPTION_WALLET_LOCATION =
(SOURCE =
  (METHOD = HSM)
  (METHOD_DATA =
    (DIRECTORY = /u01/app/oracle/admin/orcl/hsmwallet)
  )
)
```

3. The wallet can be migrated, when the server is configured:

›_ Console

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "pin[|<token_label>]";

ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "pin[|<token_label>]" [MIGRATE
USING sw_wallet_password]
```

6.4 Setting up Auto-Open for HSM Wallet

In order for the HSM Wallet to open automatically at the Oracle DB server start, an auto-open software wallet needs to be created, which has to contain the password and slot name of the HSM wallet.

This process describes the setup of auto login on the existing HSM wallet that was not upgraded by the Oracle Wallet software. For other scenarios, please see the Transparent Data Encryption Best Practices

1. Change the **sqlnet.ora** file.
2. Add a path to auto-login software wallet that will be created later in this process (**METHOD DATA**):

ENCRYPTION_WALLET_LOCATION=
(SOURCE =
(METHOD = HSM)
(METHOD_DATA =
(DIRECTORY = /u01/app/oracle/admin/orcl/hsmwallet)
)
)

3. Create a (local) auto-login wallet.

Local auto open wallets cannot be transferred to other devices.

›_ Console

```
# cd /u01/app/oracle/admin/orcl/

# mkdir hsmwallet

# cd hsmwallet
```

4. Create the auto open option.

›_ Console

```
# orapki wallet create -wallet . -auto_login[_local]
```

5. Add the following entry to the empty wallets to enable an ‘auto-open’ HSM:

›_ Console

```
# mkstore -wrl . -createEntry ORACLE.TDE.HSM.AUTOLOGIN AnyText
```

6. Insert the HSM password and the slot name to auto open the wallet.
    a. Close the connection to the HSM and open it with:

›_ Console

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "1234|
LinuxOracle19c";

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "1234|
LinuxOracle19c";
```

This will insert **HSM_auth_string[|<slot_name>]** into the auto-open wallet. From now on, no password is required to access the encrypted data with the TDE master encryption key, stored in the HSM.

7. Restart the DB server.

›_ Console

```
SQL> shutdown immediate

SQL> startup
```

8. Test if the HSM wallet is open.

›_ Console

```
SQL> CONNECT user2/Pwd01

SQL> SELECT * FROM cust_payment_info
```

- If data is returned, auto-open wallet was setup successfully.

If no data is in the table the test is not valid, since the wallet is not used!

6.5 Using TDE to Secure the Data

6.5.1 Encrypting Tablespace

When an entire tablespace is encrypted, all the data that is stored in this tablespace will be encrypted, so any table created in this tablespace will be encrypted as well.

Here is an example, where the first such tablespace is created, then a table is created and a row is inserted.

›_ Console

```
SQL> CREATE TABLESPACE securespace

  DATAFILE '/u01/app/oracle/product/12.1.0.2/db_1/secure01.dbf'

  SIZE 150M

  ENCRYPTION

  DEFAULT STORAGE(ENCRYPT); Tablespace created.

SQL> alter session set "_ORACLE_SCRIPT"=true;

SQL> CREATE USER user1 IDENTIFIED BY Pwd01 DEFAULT TABLESPACE notsecurespace;

SQL> GRANT DBA To user1;

SQL> CONNECT user1/Pwd01

SQL> CREATE TABLE cust_payment_info

  (first_name VARCHAR2(11),    last_name VARCHAR2(10),    order_number NUMBER(5)
,    credit_card_number VARCHAR2(16) ENCRYPT NO SALT,   active_card VARCHAR2(3))
;
```

1. Insert one row:

›_ Console

```
SQL> INSERT INTO cust_payment_info

  (first_name, last_name, order_number, credit_card_number, active_card)

  VALUES ('John', 'White', 22, '2643282394', 'ACT');
```

2. Get the data from the table:

›_ Console

```
SELECT * FROM cust_payment_info;
```

6.5.2 Using DBMS_CRYPTO

DBMS_CRYPTO does not support storing keys in a wallet by itself, but a table can be used for storing these keys that are either stored in an encrypted tablespace or have an encrypted key column.

1. Create a table for storing the keys and add a key to the table:

›_ Console

```
SQL > CREATE TABLE secured_keys (key_name VARCHAR2(30), key RAW(32) ENCRYPT NO
SALT);

SQL > INSERT INTO secured_keys (key_name, key) VALUES ('TestKey',

DBMS_CRYPTO.RANDOMBYTES(32));
```

2. Save the following script to a file:

```
DECLARE
input_string VARCHAR2 (200) := 'Secret Message';
output_string VARCHAR2 (200);
encrypted_raw RAW (2000); -- stores encrypted binary text
decrypted_raw RAW (2000); -- stores decrypted binary text
key_bytes_raw RAW (32); -- stores 256-bit encryption key
encryption_type PLS_INTEGER := -- total encryption type
DBMS_CRYPTO.ENCRYPT_AES256
+ DBMS_CRYPTO.CHAIN_CBC
+ DBMS_CRYPTO.PAD_PKCS5;
BEGIN
DBMS_OUTPUT.PUT_LINE ('Original string: ' || input_string);
SELECT key INTO key_bytes_raw FROM secured_keys WHERE key_name = 'TestKey';
encrypted_raw := DBMS_CRYPTO.ENCRYPT
(
src => UTL_I18N.STRING_TO_RAW (input_string, 'AL32UTF8'),
typ => encryption_type,
key => key_bytes_raw
);
-- The encrypted value "encrypted_raw" can be used here
DBMS_OUTPUT.PUT_LINE ('Encrypted: ' || RAWTOHEX(encrypted_raw));
decrypted_raw := DBMS_CRYPTO.DECRYPT
(
src => encrypted_raw,
typ => encryption_type,
key => key_bytes_raw
);
output_string := UTL_I18N.RAW_TO_CHAR (decrypted_raw, 'AL32UTF8');
DBMS_OUTPUT.PUT_LINE ('Decrypted string: ' || output_string);
END;
```

Alternatively, the following two commands can be used instead of the script:

SQL > CONNECT sqlplus as sysdba;

SQL > GRANT EXECUTE ON sys.dbms_crypto TO user2;

›_ Console

```
SQL > INSERT INTO secured_keys (key_name, key) VALUES ('TestKey',

DBMS_CRYPTO.RANDOMBYTES(32));
```

3. Enable the server output and run the script:

›_ Console

```
SQL> SET SERVEROUTPUT ON

SQL> \(filename)

SQL> \
```

6.5.3 Encrypting Columns in Tables

In order to create a table with an encrypted column, the ENCRYPT keyword needs to be added, when specifying the COLUMN. Any user with an access to the encrypted columns can read the data from these columns, but the data is encrypted on the disk.

Example:

›_ Console

```
SQL> CREATE TABLESPACE notsecurespace

  DATAFILE '/u01/app/oracle/product/12.1.0.2/db_1/notsecure01.dbf'   SIZE 150M;

Tablespace created.

SQL> alter session set "_ORACLE_SCRIPT"=true;

SQL> CREATE USER user2 IDENTIFIED BY Pwd01 DEFAULT TABLESPACE notsecurespace;

SQL> GRANT DBA To user2;

SQL> CONNECT user2/Pwd01

SQL> CREATE TABLE cust_payment_info

  (first_name VARCHAR2(11),    last_name VARCHAR2(10),    order_number NUMBER(5)
,    credit_card_number VARCHAR2(16) ENCRYPT NO SALT,   active_card VARCHAR2(3))
;
```

2. Insert one row:

›_ Console

```
SQL> INSERT INTO cust_payment_info

  (first_name, last_name, order_number, credit_card_number, active_card)

  VALUES ('John', 'White', 22, '2643282394', 'ACT');
```

1. Indexes can be created on the encrypted columns:

›_ Console

```
SQL> CREATE INDEX cust_payment_info_idx ON cust_payment_info
(credit_card_number);
```

If a column in an existing table needs to be encrypted, the table has to be altered:

›_ Console

```
SQL> ALTER TABLE cust_payment_info MODIFY (first_name VARCHAR2(11) ENCRYPT);
```

7 Backup and Restore

The Utimaco HSM enables different ways of making backups of either the entire database of keys or just groups of keys. All the backups are encrypted by using the Master Backup Key (MBK), generated by the system administrator, when setting up the HSM. More information about the MBK can be found in [CSADMIN].

7.1 Backing up and Restoring Key Database

All of the keys inside the HSM are stored in a CXI database and it is possible to backup the entire database at once. In the same way, the user database can be backed up as well. In FIPS mode this feature is not supported.

1. Open the Crypto Administration Tool.
2. Make sure that the HSM is connected and in the operational mode.
3. Click on **Login/Logoff** to open the **Login/Logoff User** window.
4. Login the appropriate users to achieve the permission level of at least 22000000.
5. Click on Backup/Restore to open the CryptoServer Database Backup/Restore Wizard window.
    a. In the Command section select either to:
        i. Backup databases from source CryptoServer to backup directory,
        ii. Restore databases from backup directory to target CryptoServer,
        iii. Copy databases from source CryptoServer to target CryptoServer.
    b. In the Settings section:
        i. Select the appropriate Source CryptoServer,
        ii. If available select the appropriate Target CryptoServer,
        iii. In the Backup directory section type the appropriate backup directory path (set to C:\Program Files\Utimaco\CryptoServer\Administration as default), or click … to browse for the appropriate directory.
6. Select the databases to backup or restore.
7. Click Execute.
8. A confirmation window appears.

For a FIPS backup, please use the P11CAT or the P11tool2 tools

7.2 Backing up and Restoring a Key Database with P11CAT

It is possible to backup separate PKCS#11 slots by using either the P11CAT or the p11tool2. In this guide we use the P11CAT for the backup procedure. Please refer to [CSP11TOOL2] for the backup procedure with the p11tool2.

1. Open the PKCS#11 CryptoServer Administration (P11CAT).
2. Login to the slot you wish to backup as the **Cryptographic User** (achieve the permission level of at least 00000002).
3. Click on **Backup/Restore**.
4. Click on **Backup/Restore Keys**.
5. Select one among the 4 options. Click the one that corresponds to your case. The possibilities are to perform either:
    a. Backup Internal Keys,
    b. Backup External Keys,
    c. Restore Key Backup to Internal Key Store,
    d. Restore Key Backup to External Key Store.
6. A popup window opens.
    a. Select the directory where the key database will be backed up and type the name of the key database in the section File name.
    b. If you chose to restore a key backup to an internal or an external key store, select the directory, where your backup is located.
    c. To confirm your choice, click on Save.
7. In the **Status** window a log of the performed action is displayed.

8 FIPS Requirements

All the steps are indentical for the HSM in FIPS 140-2 approved mode. The only difference is that the backup of the entire key database is not possible. In this case the P11CAT or the p11tool2 are used for backing up the keys.

Note that although the integration does not require extra steps, the HSM running in FIPS mode will accept ONLY FIPS compliant parameters. Be careful to select FIPS compliant algorithms, key lengths and elliptic curves when generating new keys. For more information about the FIPS compliant algorithms please refer to the CryptoServer User and Administration Guides.

9 Further Information

This document forms a part of the information and support which is provided by the Utimaco IS GmbH company. Additional documentation can be found on the product CD in the Documentation directory.

All CryptoServer product documentation is also available at the Utimaco IS GmbH website:

http://hsm.utimaco.com

10 References

Reference	*Title/Company*	*Document No.*
[CSADMIN]	CryptoServer – csadm Manual/Utimaco IS GmbH	2009-0003
[OraDASAG]	Oracle Help Center - Securing Stored Data Using Transparent Data Encryption	
[OBOD19c]	Oracle Database 19c Installation On Oracle Linux v7 (OL7)	
[TdeBestPr]	Transparent Data Encryption Best Practices	
[CSPKCSDG]	CryptoServer - PKCS#11 R2 Developer Guide	2012-0007
[CSPKCSM]	CryptoServer - PKCS#11 P11CAT Manual	M013-0001-en
[LPKCSHD]	Learning PKCS#11 in Half a Day	2015-0008
[CSP11TOOL2]	CryptoServer PKCS#11 p11tool2 Reference Manual	2012-0004

Contact

Utimaco IS GmbH

Germanusstr. 4

D-52080 Aachen Germany

Phone: +49 241 1696 – 200

Fax: +49 241 1696 – 199

Web: http://hsm.utimaco.com

E-mail: hsm@utimaco.com