ORACLE DATABASE 19C TRANSPARENT DATA ENCRYPTION

Integration Guide

Applicable Devices:
KMES Series 3

THIS DOCUMENT CONTAINS CONFIDENTIAL INFORMATION PROPRIETARY TO FUTUREX, LP. ANY UNAUTHORIZED USE, DISCLOSURE, OR DUPLICATION OF THIS DOCUMENT OR ANY OF ITS CONTENTS IS EXPRESSLY PROHIBITED.

TABLE OF CONTENTS

[1] DOCUMENT INFORMATION

[1.1] DOCUMENT OVERVIEW

This document provides information about the integration of the Futurex KMES Series 3 with Oracle Database 19c Transparent Data Encryption (TDE) using Futurex PKCS #11 libraries. For additional questions related to the KMES Series 3, see the relevant user guide.

[1.1.1] ABOUT ORACLE TDE

From the Oracle documentation website: “Transparent data encryption enables you to encrypt sensitive data, such as credit card numbers, stored in table columns. Encrypted data is transparently decrypted for a database user who has access to the data. Transparent data encryption helps protect data stored on media in the event that the storage media or data file gets stolen.”

[2] ORACLE DATABASE 19C TRANSPARENT DATA ENCRYPTION (TDE) INTEGRATION OVERVIEW

Integrating Oracle Database 19c Transparent Data Encryption (TDE) with the KMES Series 3 requires the Futurex PKCS #11 (FXPKCS11) library. Once configured, the Master Encryption Key (MEK) used for TDE can be stored within the confines of a FIPS 140-2 Level 3 validated HSM (i.e., the KMES Series 3), adding a layer of protection for data-at-rest.

The Master Encryption Key encrypts the Oracle Table Keys, which encrypt or decrypt columns or tablespaces locally in the database. Each table has its own table key. From the client application perspective, the encryption and decryption process is transparent, so there is no need to make any changes to the existing application.

The connection between the Futurex PKCS #11 library and the KMES Series 3 must be a mutually authenticated TLS connection. Therefore, TLS/SSL certificates must be created (which will be done using OpenSSL and a CA on the KMES itself) to provide certificates for both the KMES Host API connection pair and the Oracle Database instance where the FXPKCS11 library is running. This guide is intended to provide the user with the required information to configure Futurex PKCS #11 with Oracle Database so that the TDE Master Encryption Key can be generated and stored on the KMES Series 3 to be used for encrypting the Oracle Table Keys. This process is summarized in the following image:

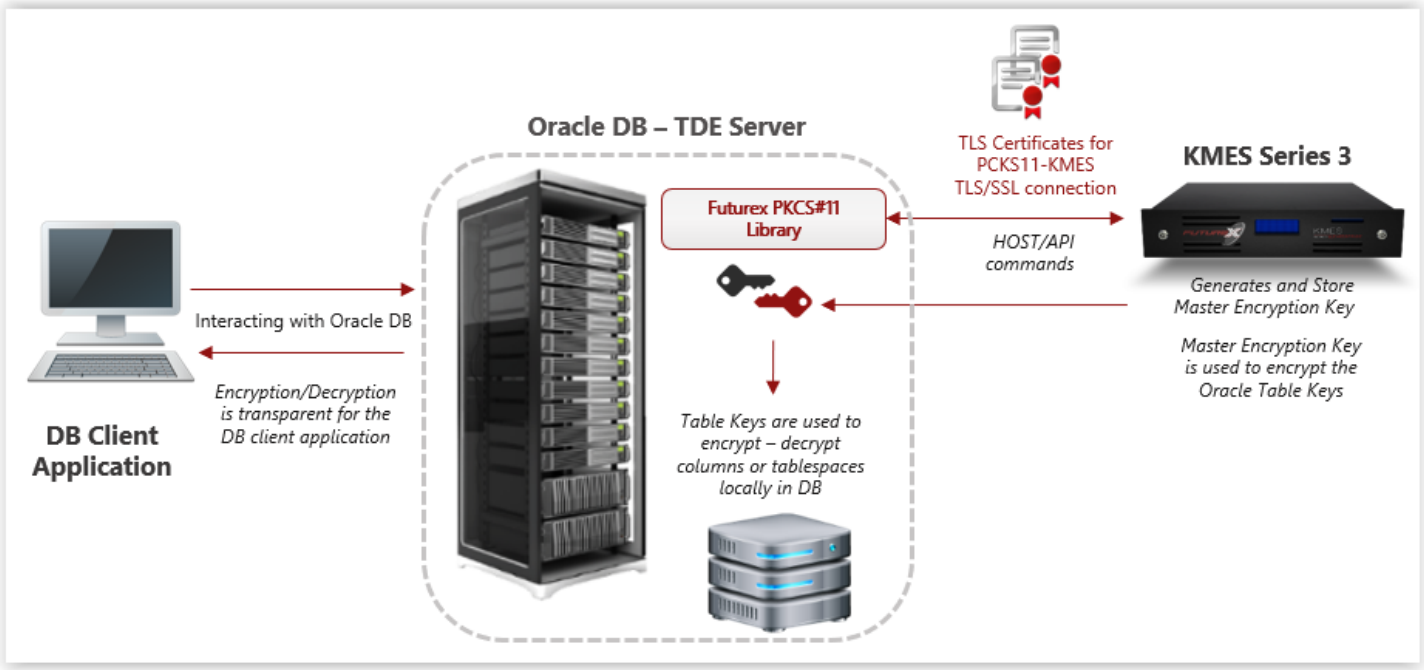


FIGURE: PROCESS OF ORACLE TDE INTEGRATION

[3] PREREQUISITES

1. Linux Server (Rocky Linux 8 for this example – GUI not required): https://rockylinux.org/download/
2. OpenSSL for Linux
3. Oracle Database 19c
4. Installation Guide for Oracle Database 19c: https://docs.oracle.com/en/database/oracle/oracle-database/19/index.html
5. Oracle Guide to securing data with Oracle TDE (General concepts – how to enable TDE): https://docs.oracle.com/en/database/oracle/oracle-database/19/asoag/introduction-to-transparent-data-encryption.html
6. Oracle Guide to use HSM with TDE: https://docs.oracle.com/en/database/oracle/oracle-database/19/asoag/configuring-transparent-data-encryption.html#GUID-753C4808-CC51-4DA1-A5C3-980417FDAB14
7. KMES Series 3 with minimum software version 6.1.2.4–ck6a85 and Futurex-signed TLS certificates preloaded
8. TLS client certificates to allow the Futurex PKCS #11 (FXPKCS11) library to connect with the KMES Series 3. Refer to the section on the KMES Series 3 TLS certificates and FXPKCS11 certificates and the section on the FXPKCS11 configuration file for additional details.
9. Futurex PKCS #11 (FXPKCS11) library version 4.34 or above

Note: For the purposes of this document, we assume that the Linux server is running, the user has root permissions, and has installed the Oracle database with the necessary configurations to support TDE beforehand.

This guide shows users how to configure the Futurex PKCS #11 library to be used as an interface for Oracle TDE to connect to a KMES Series 3 HSM wallet, based on: https://docs.oracle.com/en/database/oracle/oracle-database/19/asoag/configuring-transparent-data-encryption.html

[4] CONFIGURING THE FUTUREX PKCS #11 LIBRARY IN ORACLE

Note: If you plan to run Oracle Database in a Docker container, skip this section. A later section covers the steps to configure the Futurex PKCS #11 library specific to a container implementation.

[4.1] COPY THE FUTUREX PKCS #11 LIBRARY TO THE REQUIRED PATH

1. Copy the Futurex PKCS #11 library file (i.e., **libfxpkcs11.so**) to the path **/opt/oracle/extapi/[32,64]/hsm/futurex/X.X/** where X.X is the library version.
2. Copy the **PKCS11Manager** and **fxpkcs11.cfg** files into the **/etc** directory.

[5] KMES SERIES 3 CONFIGURATION

The first half of this section covers general KMES Series 3 configurations you must make for Oracle Database to store the TDE Master Encryption Key on the KMES. The second half of this section covers the steps to configure TLS communication between the KMES Series 3 and the Oracle Database instance.

[5.1] CRYPTO OPERATOR GROUP AND USERS

Users must create a new group (as an example: **Crypto Operators** which will include a **Crypto1** user and their credentials). The Futurex PKCS #11 library can access the KMES Series 3 with any user created within the group.

Note: The **Crypto1** user will later be configured in the fxpkcs11.cfg file to allow the FXPKCS11 library to connect to the KMES Series 3.

To define the **Crypto Operators** group and user(s), complete the following sequence:

1. Log in to KMES Series 3 with a user that has privileges to Add/Modify user groups.
2. Select **Users** > **Admin Group**. Right-click **Add Group**.
3. Set:
    - **Group name**to **CryptoOperators**
    - **Number of users required to log in** to **1**
    - **User type** to **Normal Users**
    - **Members can Authenticate to** to **Client**, **Host API**
4. Select the following permissions:
    - **Manage Certificates** > **All**
    - **Manage Keys** > **All**
    - **Manage Encryption Device Groups** > **All**
    - **Perform Cryptographic operations** > **All**
5. Go to **Users** > **CryptoOperator Group.** Right-click **Add User**.
6. Enter a username (**Crypto1** as an example).
7. Enter a password for this user (**safest** as an example). This password will be required later.

[5.2] CREATE THE KEY GROUP FOR ORACLE TDE KEYS

A key group must be created for the Oracle TDE – KMES Series 3 integration. This key group will contain the created or renewed Master Keys for Oracle TDE.

Note: The name of the key group can be anything, but remember the name because it will need to be configured later inside the **<KEYGROUP-NAME>** tag in the fxpkcs11.cfg file.

To create the key group, complete the following steps:

1. Log in to KMES Series 3 with a user with privileges to Add/Modify key groups.
2. Select **Keys**. Right-click **Add Key Group.**

3. Set the following details:
    - **Name** to **OracleTDE** (as an example)
    - **OwnerGroup** to **CryptoOperators** (the name of the previously created user group)
    - **Permissions** to the following:
        - **AdminGroup**: **Add**, **implicit**
        - **CryptoOperator**: **Add**

The crypto user (**Crypto1** in this example) and the key group name (**OracleTDE** in this example) will be required in following fields in the fxpkcs11.cfg file:

```
# KMES in first slot
<KMS>
    # Which PKCS11 slot
    <SLOT>              0               </SLOT>

    # Login username
    <CRYPTO-OPR>        Crypto1         </CRYPTO-OPR>

    # Key group name
    <KEYGROUP-NAME>     OracleTDE       </KEYGROUP-NAME>
```

[5.3] ENABLE THE HOST API COMMANDS REQUIRED FOR THE ORACLE TDE OPERATION

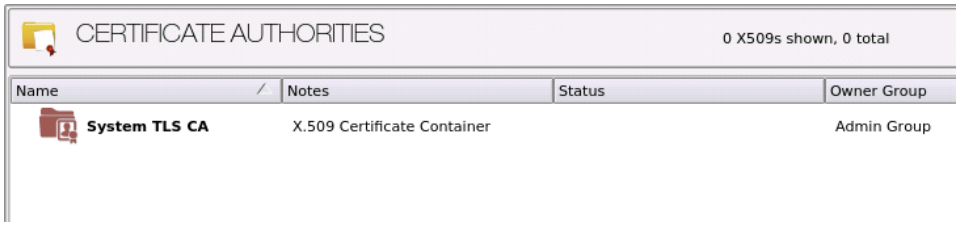
Because the connection to the FXPKCS11 library will use the Host API port, users must define which commands are enabled for execution by the FXPKCS11 library. To set the enabled commands, complete the following steps:

1. Log in to KMES Series 3 with privileges to modify Host API configuration.
2. Select **Configuration** > **Host API Options**, enable the commands listed below:
    - **ECHO**: Communication Test/Retrieve Version
    - **RAFA**: Filter Issuance Policy
    - **RKCK**: Create Key
    - **RKCP**: Get Command Permissions
    - **RKCS**: Create Symmetric Key Group
    - **RKED**: Encrypt or Decrypt Data
    - **RKLN**: Lookup Objects
    - **RKLO**: Login User
    - **RKRC**: Get Key
3. Select **Save**

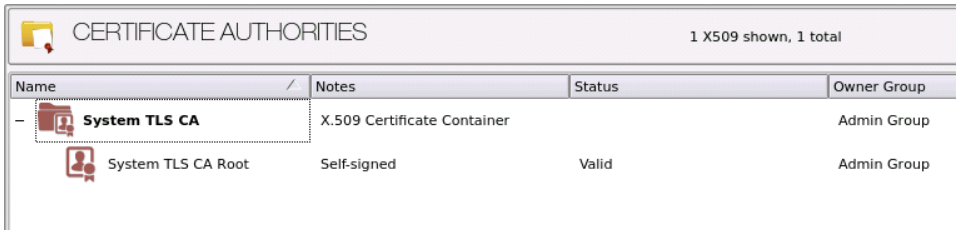
[5.4] CONFIGURE TLS COMMUNICATION BETWEEN THE KMES SERIES 3 AND THE ORACLE DATABASE INSTANCE

[5.4.1] Create a Certificate Authority (CA)

1. Log in to the KMES Series 3 application interface with the default Admin identities.
2. Select **Certificate Authorities** in the left menu, then select **Add CA....** at the bottom of the page.
3. In the **Certificate Authority** dialog, enter a name for the Certificate Container, leave all other fields as the default values, and select **OK**.
4. The **Certificate Container** that was just created should appear in the Certificate Authorities menu.



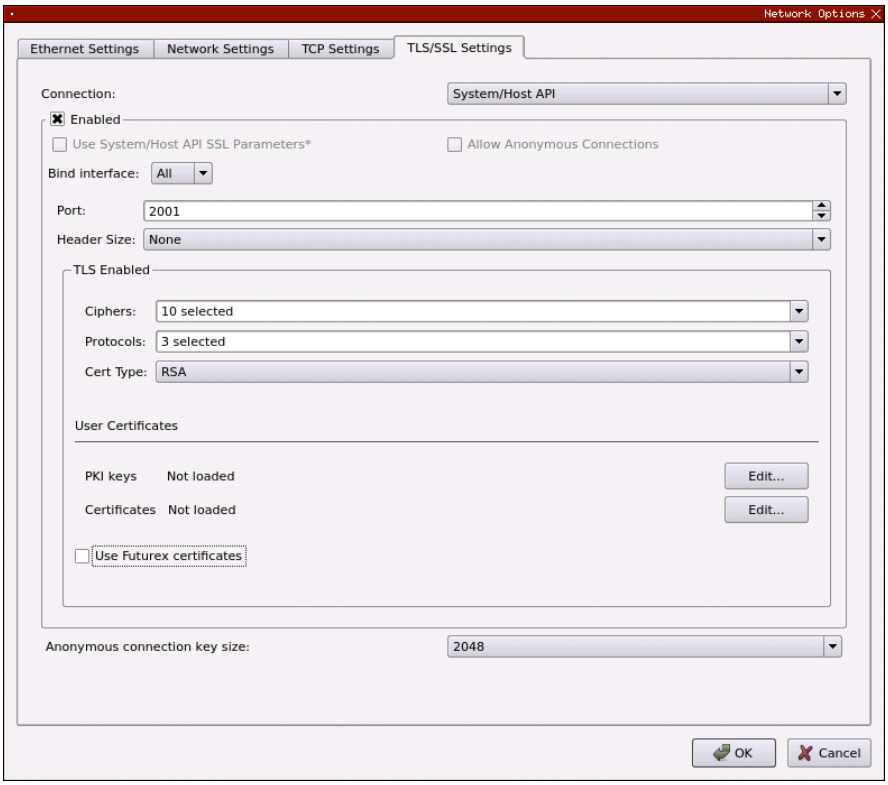
5. Right-click on the **Certificate Container** and select **Edit...**. In the **Certificate Authority** dialog, check the box that says **Can be used for PKI authentication**, and select **OK** to save.
6. Right-click on the **Certificate Container** again and select **Add Certificate** > **New Certificate...**
7. In the **Subject DN** tab, set a common name for the certificate, such as **System TLS CA Root**.
8. In the **Basic Info** tab, change the Major key to the **PMK**. You can leave all other settings as the default values.
9. In the **V3 Extensions** tab, select the **Example Certificate Authority** profile, and select **OK**.
10. The root CA certificate is now listed under the previously created **Certificate Container**.



[5.4.2] Generate a CSR for the System/Host API connection pair

1. Go to **Configuration** > **Network Options**.
2. In the **Network Options** dialog, select **TLS/SSL Settings**.

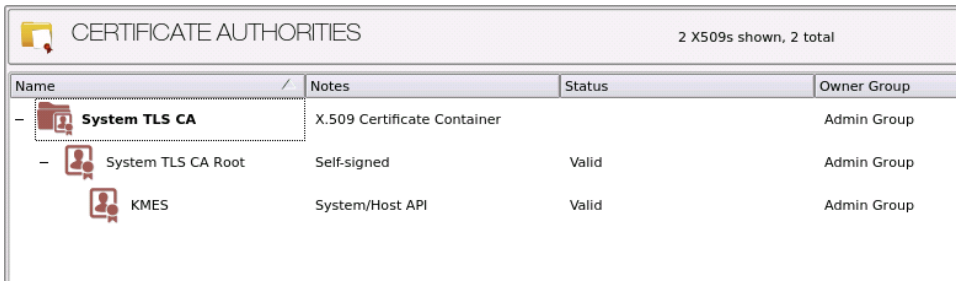
3. Under the **System/Host API** connection pair, uncheck **Use Futurex certificates** and select **Edit...** next to PKI keys.



4. In the **Application Public Keys** dialog, select **Generate...**
5. A warning will appear stating that SSL will not be functional until you import new certificates. Select **Yes** to continue.
6. In the **PKI Parameters** dialog, change the Encrypting key to the **PMK** and the Key Size to **2048**, then select **OK**.
7. The PKI Key Pair should load in the **Application Public Keys** dialog. Select **Request...**
8. In the **Subject DN** tab, set a common name for the certificate, such as **KMES**.
9. In the **V3 Extensions** tab, select the **Example TLS Server Certificate** profile.
10. In the **PKCS #10 Info** tab, select a save location for the CSR and select **OK**.
11. A message should appear stating that the certificate signing request was successfully written to the selected file location. Select **OK**.
12. Select **OK** to save the **Application Public Keys** settings.
13. In the main **Network Options** dialog, it should now say **Loaded** next to **PKI keys** for the System/Host API connection pair.

[5.4.3] Sign the System/Host API CSR

1. Open the **Certificate Authorities** menu.
2. Right-click on the **System TLS CA Root** certificate and select **Add Certificate** > **From Request...**
3. In the file browser, find and select the CSR for the System/Host API connection pair.
4. You don't need to modify any of the settings for the certificate once loaded. Select **OK**.
5. The signed System/Host API certificate should show under the root CA certificate on the **Certificate Authorities** page.



[5.4.4] Export the Root CA certificate

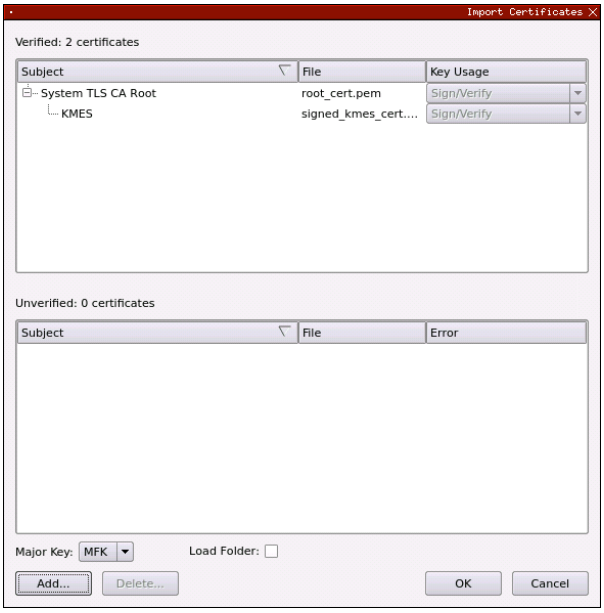
1. Open the **Certificate Authorities** menu.
2. Right-click on the **System TLS CA Root** certificate and select **Export** > **Certificate(s)...**
3. In the **Export Certificate** dialog, change the encoding to **PEM**, then select **Browse...**
4. In the file browser, select the location where you want to save the root CA Certificate. Specify **tls_ca.pem** as the name for the file and select **Open**.
5. Select **OK**. A message box should open stating that the PEM file was successfully written to the location you specified.

[5.4.5] Export the signed System/Host API certificate

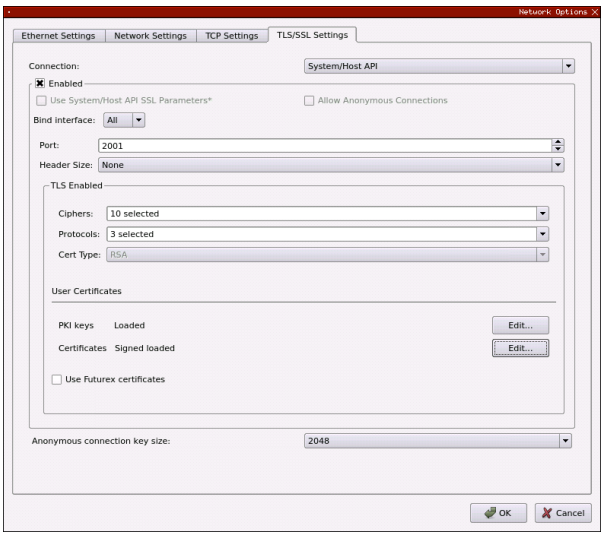
1. Open the **Certificate Authorities** menu.
2. Right-click on the **KMES** certificate and select **Export** > **Certificate(s)...**
3. In the **Export Certificate** dialog, change the encoding to **PEM** and select **Browse...**
4. In the file browser, select the location where you want to save the signed System/Host API certificate. Specify a name for the file, then click **Open**.
5. Select **OK**. A message box should pop up stating that the PEM file was successfully written to the location that you specified.

[5.4.6] Load the exported certificates into the System/Host API connection pair

1. Go to **Configuration** > **Network Options**.
2. In the **Network Options** dialog, select **TLS/SSL Settings**.
3. Select **Edit...** next to **Certificates** in the **User Certificates** section.
4. Right-click on the **System/Host API SSL CA** X.509 Certificate Container, then select **Import...**
5. Click **Add...** at the bottom of the **Import Certificates** dialog.
6. In the file browser, select both the root CA certificate and the signed System/Host API certificate, then select **Open**. The certificate chain should appear as shown below:



7. Select **OK** to save changes. In the **Network Options** dialog, the **System/Host API** connection pair should say Signed loaded next to **Certificates** in the **User Certificates** section, as shown below:



8. Select **OK** to save and exit the Network Options dialog.

[5.4.7] Generate a private key and certificate signing request for the Oracle Database instance using OpenSSL

Note: You must run the commands in this section from a terminal application that has OpenSSL installed.

1. Open a terminal and run the following command to generate a private key for the Oracle Database instance:

   ```
   $ openssl genrsa -out tls_skey.pem 2048
   ```

   The private key will output to a file named tls_skey.pem in the current working directory.

2. Run the following command to generate a Certificate Signing Request (CSR) for the Oracle Database instance:

   ```
   $ openssl req -new -key tls_skey.pem -out tls_cert_req.pem -days 365
   ```

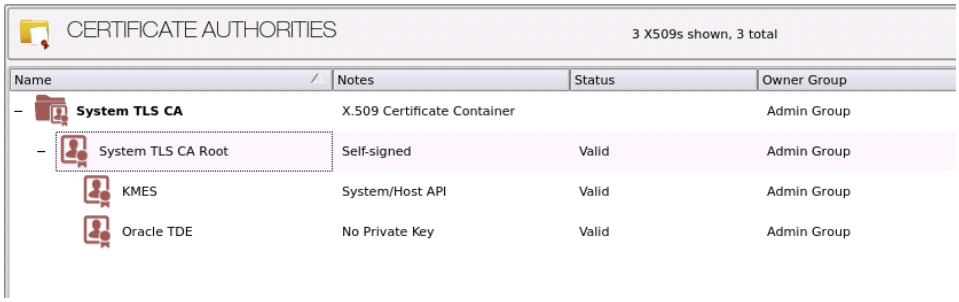
   The command will prompt you to enter certificate information. Press **Enter** at every prompt to set the default value for every field.

   The CSR will output to a file named tls_cert_req.pem in the current working directory.

3. Move or copy the CSR file (i.e., tls_cert_req.pem) to the storage medium configured on the KMES.

[5.4.8] Sign the Certificate Signing Request (CSR) for the Oracle Database instance

1. Open the **Certificate Authorities** menu.
2. Right-click on the **System TLS CA Root** certificate and select **Add Certificate** -> **From Request...**
3. In the file browser, find and select the Oracle Database CSR. Certificate information will populate in the **Create X.509 From CSR** window.
4. In the **Subject DN** tab, change the preset dropdown to **Classic**, and set a common name for the certificate, such as **Oracle TDE**.
5. All settings in the **Basic Info** tab can be left as the default values.
6. In the **V3 Extensions** tab, select the **Example TLS Client Certificate** profile and select **OK**.
7. The signed **Oracle TDE** certificate is listed now under the **System TLS CA Root** certificate.



[5.4.9] Export the signed Oracle TDE certificate

1. Open the **Certificate Authorities** menu.
2. Right-click on the **Oracle TDE** certificate and select **Export** -> **Certificate(s)...**
3. In the **Export Certificate** dialog, change the encoding to **PEM** and select **Browse...**
4. In the file browser, navigate to the location where you want to save the Oracle TDE certificate. Specify **tls_cert.pem** as the name for the file, and select **Open**.
5. Select **OK**. A message box should appear stating that the PEM file was successfully written to the location you specified.

Note: You must move both the **Oracle TDE** certificate and the **System TLS CA Root** certificate to the computer running the Oracle Database instance. In the next section, the certificates will be configured inside the Futurex PKCS #11 configuration file and used for TLS communication with the KMES Series 3.

[6] EDIT THE FUTUREX PKCS #11 CONFIGURATION FILE

Note: If you plan to run Oracle Database in a Docker container, skip this section. A later section covers the steps to configure the Futurex PKCS #11 configuration file specific to a container implementation.

The Futurex PKCS #11 configuration file (i.e., fxpkcs11.cfg) is used by the Futurex PKCS #11 library to connect to the KMES. It enables the user to modify certain configurations and set connection details. This section covers the **<KMS>** portion of the FXPKCS11 config file, where the connection details are set.

Note: By default, the FXPKCS11 library looks for the configuration file at C:\Program Files\Futurex\fxpkcs11\fxpkcs11.cfg for Windows and /etc/fxpkcs11.cfg for Linux. Alternatively, the FXPKCS11_CFG environment variable can be set to the location of the fxpkcs11.cfg file.

Open the fxpkcs11.cfg file in a text editor as an administrator and edit it accordingly.

```
<KMS>
    # Which PKCS11 slot
    <SLOT>                  0                       </SLOT>
    <LABEL>                 Futurex                 </LABEL>

    # HSM crypto operator user name
    <CRYPTO-OPR>            Crypto1                 </CRYPTO-OPR>

    # Key group name
    <KEYGROUP-NAME>         OracleTDE               </KEYGROUP-NAME>

    # Connection information
    <ADDRESS>               10.0.8.30               </ADDRESS>
    <PROD-PORT>             2001                    </PROD-PORT>
    <PROD-TLS-ENABLED>      YES                     </PROD-TLS-ENABLED>
    <PROD-TLS-ANONYMOUS>    NO                      </PROD-TLS-ANONYMOUS>
    <PROD-TLS-CA>           /home/oracle/pki/tls_ca.pem       </PROD-TLS-CA>
    <PROD-TLS-CERT>         /home/oracle/pki/tls_cert.pem     </PROD-TLS-CERT>
    <PROD-TLS-KEY>          /home/oracle/pki/tls_skey.pem     </PROD-TLS-KEY>
    #<PROD-TLS-KEY-PASS>     safest                 </PROD-TLS-KEY-PASS>

    # YES = This is communicating through a Guardian
    <FX-LOAD-BALANCE>       NO                      </FX-LOAD-BALANCE>
</KMS>
```

The **<SLOT>** and **<LABEL>** fields specify PKCS11 slot 0 and the label *Futurex*.

In the **<CRYPTO-OPR>** field, specify the name of the user you created on the KMES.

In the **<KEYGROUP-NAME>** field, specify the name of the key group that you created on the KMES.

In the **<ADDRESS>** field, specify the IP address of the KMES that the FXPKCS11 library should connect to.

In the **<PROD-PORT>** field, specify port *2001*, which is the System/Host API port on the KMES.

The **<PROD-TLS-ENABLED>** field should be set to *YES*.

The **<PROD-TLS-ANONYMOUS>** field should be set to *NO*.

In the **<PROD-TLS-CA>** field, specify the path to where the tls_ca.pem file is saved.

In the **<PROD-TLS-CERT>** field, specify the path to where the tls_cert.pem file is saved.

In the **<PROD-TLS-KEY>** field, specify the path to where the tls_skey.pem file is saved.

The **<PROD-TLS-KEY-PASS>** field should remain commented out because a password was not set for the client private key.

If you use Guardian to manage HSMs in a cluster, define the **<FX-LOAD-BALANCE>**field as *YES*. Otherwise, set it to *NO*.

After you finish editing the fxpkcs11.cfg file, run the PKCS11Manager file to test the connection against the HSM and check the fxpkcs11.log for errors and information. For more information, refer to the Futurex PKCS #11 technical reference found on the Futurex Portal.

[7] GENERATE A TDE MASTER ENCRYPTION KEY ON THE KMES SERIES 3

To configure Oracle Database 19c TDE with an HSM, we recommend that you refer to the Oracle knowledge base article below:

- Oracle Database 19c: https://docs.oracle.com/en/database/oracle/oracle-database/19/asoag/configuring-transparent-data-encryption.html

This section walks through a very basic example of configuring Oracle TDE with an HSM using PKCS #11. However, there are many nuances in an Oracle Database environment, so the following steps will not apply directly to certain situations and implementations. Use this section only as a general guide, and thoroughly consult the Oracle documentation linked above before implementing Oracle TDE with an HSM in your environment.

To use HSM-based encryption, a Master Encryption Key (MEK) must be generated, which will be stored on the KMES, and used by TDE for encrypting and decrypting the Oracle Table Keys.

Two different Oracle Database implementations are covered in this section. A standard implementation of Oracle Database running on a server or desktop, and an Oracle Database implementation running in a Docker container.

[7.1] STANDARD IMPLEMENTATION

1. Set the Oracle environment with the following commands:
   Note: The **oraenv** tool sets up the Oracle database environment for the current session and allows use of the **sqlplus** command. To set the Oracle environment, follow the command sequence below. When prompted, specify the system ID (SID) for the instance — "orcl" in this example — or use the default value indicated between the brackets in line 4 below. All instances on the system will require a unique SID.

   ```
   $ su oracle

   $ cd ~

   $ . /usr/local/bin/oraenv
   ORACLE_SID = [oracle] ? orcl
   ```

   Upon success, the command will return the following message:

   ```
   The Oracle base has been set to /home/oracle/app/oracle
   ```

2. Connect to the database:

   ```
   $ sqlplus / as sysdba
   ```

3. Start the Oracle instance:

   ```
   SQL> startup
   ```

4. Set the static **WALLET_ROOT** parameter, which allows you to designate the location of the keystore you plan to use.

   Note: You must set up the WALLET_ROOT parameter even if you do not use a keystore.

```
SQL> ALTER SYSTEM SET WALLET_ROOT = '/opt/oracle/extapi/64/hsm/futurex/4.45/libfxpkcs11.so'
SCOPE=SPFILE;
```

5. Bounce the database after setting the WALLET_ROOT parameter by shutting it down and starting it back up.

```
SQL> SHUTDOWN IMMEDIATE;

SQL> STARTUP;
```

6. Set the dynamic TDE_CONFIGURATION parameter that allows you to designate the type of keystore you plan to use.

```
SQL> ALTER SYSTEM SET TDE_CONFIGURATION='KEYSTORE_CONFIGURATION=HSM' SCOPE=BOTH SID = '*';
```

7. Bounce the database after setting the TDE_CONFIGURATION parameter by shutting it down and starting it back up.

```
SQL> SHUTDOWN IMMEDIATE;

SQL> STARTUP;
```

8. Open the hardware keystore using the password of the user created on the KMES:

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

9. Create the TDE Master Encryption Key using the password of the user created on the KMES:

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

Note: If migrating a previously configured TDE Master Encryption Key, refer to this article.

10. If successful, the following message appears.

```
System altered.
```

Keys generated on the KMES Series 3 look like the following:

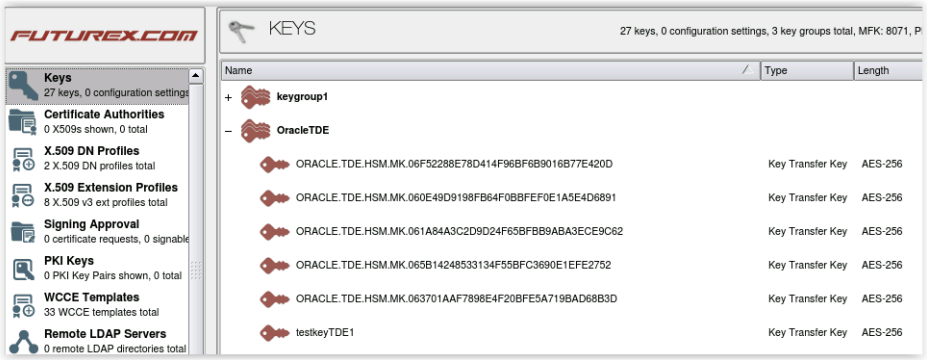


FIGURE: GENERATED KEYS IN THE KMES SERIES 3

Note: If the database contains columns encrypted with a public key, the columns are decrypted and re-encrypted with the Oracle table key, which is encrypted/decrypted with the AES symmetric key generated by HSM-based transparent data encryption.

[7.2] DOCKER CONTAINER IMPLEMENTATION

The steps outlined in this section are specific to how you must configure the Futurex PKCS #11 (FXPKCS11) library to work with an Oracle Database Docker container. For instructions on how to build and run Oracle Database in a Docker container, please refer to Oracle's documentation.

1. On the host computer that will be running the Oracle Database container, open a terminal and navigate to the location where the Oracle Database private key (tls_skey.pem) is saved.
2. Run the following command to make the Oracle Database private key readable and executable for all users:

```
$ chmod 555 tls_skey.pem
```

3. Set the OpenSSL version to match your container in the **OPENSSL_VERSION** environment variable. If running Oracle Database 19c using the official Oracle Database container images repository on GitHub (https://github.com/oracle/docker-images/blob/main/OracleDatabase), you need to run the following command because that container image is based on Oracle Linux 7, which is OpenSSL 1.0-based:

```
$ OPENSSL_VERSION=OpenSSL-1.0.x
```

   If your Oracle Database runs in a container based on OpenSSL 1.1, run the following command:

```
$ OPENSSL_VERSION=OpenSSL-1.1.x
```

4. Download the Futurex PKCS #11 (FXPKCS11) library from the **Futurex Portal**. If your container is based on OpenSSL 1.0, download the fxpkcs11-redhat-4.xx-xxxx.tar file. If your container is based on OpenSSL 1.1, download the fxpkcs11-redhat8-4.xx-xxxx.tar file.
5. Extract the FXPKCS11 library and save the version in the **PKCS_VERSION** environment variable.

```
$ tar -xvf fxpkcs11*.tar
$ PKCS_VERSION=$(grep -r --include=*info* Version: | awk 'NR==2{print $2}')
```

6. Edit the fxpkcs11.cfg file to allow the FXPKCS11 library to connect to the KMES Series 3. Set the fields shown below inside the **<KMS>** section:

```
<KMS>
    # Which PKCS11 slot
    <SLOT>                  0                       </SLOT>
    <LABEL>                 Futurex                 </LABEL>

    # HSM crypto operator user name
    <CRYPTO-OPR>            Crypto1                 </CRYPTO-OPR>

    # Key group name
    <KEYGROUP-NAME>         OracleTDE               </KEYGROUP-NAME>

    # Connection information
    <ADDRESS>               10.0.8.30               </ADDRESS>
    <PROD-PORT>             2001                    </PROD-PORT>
    <PROD-TLS-ENABLED>      YES                     </PROD-TLS-ENABLED>
    <PROD-TLS-ANONYMOUS>    NO                      </PROD-TLS-ANONYMOUS>
    <PROD-TLS-CA>           /home/oracle/pki/tls_ca.pem       </PROD-TLS-CA>
    <PROD-TLS-CERT>         /home/oracle/pki/tls_cert.pem     </PROD-TLS-CERT>
```

```
    <PROD-TLS-KEY>          /home/oracle/pki/tls_skey.pem     </PROD-TLS-KEY>
    #<PROD-TLS-KEY-PASS>     safest                  </PROD-TLS-KEY-PASS>

    # YES = This is communicating through a Guardian
    <FX-LOAD-BALANCE>       NO                      </FX-LOAD-BALANCE>
</KMS>
```

The **<SLOT>** and **<LABEL>** fields specify PKCS11 slot 0 and the label *Futurex*.

In the **<CRYPTO-OPR>** field, specify the name of the user you created on the KMES.

In the **<KEYGROUP-NAME>** field, specify the name of the key group that you created on the KMES.

In the **<ADDRESS>** field, specify the IP address of the KMES that the FXPKCS11 library should connect to.

In the **<PROD-PORT>** field, specify port *2001*, which is the System/Host API port on the KMES.

The **<PROD-TLS-ENABLED>** field should be set to *YES*.

The **<PROD-TLS-ANONYMOUS>** field should be set to *NO*.

In the **<PROD-TLS-CA>** field, specify the path to where the tls_ca.pem file is saved.

In the **<PROD-TLS-CERT>** field, specify the path to where the tls_cert.pem file is saved.

In the **<PROD-TLS-KEY>** field, specify the path to where the tls_skey.pem file is saved.

The **<PROD-TLS-KEY-PASS>** field should remain commented out because a password was not set for the client private key.

Set **<FX-LOAD-BALANCE>**to the default value *NO*.

7. Run the following command to start the Oracle Database container and bind-mount all of the FXPKCS11 files needed for FXPKCS11 to be able to connect to the KMES Series 3.

   Note: This command needs to be run from the same directory where the extracted **fxpkcs11** directory is stored.

   Note: If the TLS certificates for authentication with the KMES Series 3 are not stored in the /home/oracle/pki directory on your system, modify the third **-v** flag in your command to reflect this.

```
$ docker run -d \
    -v $(pwd)/fxpkcs11.cfg:/etc/fxpkcs11.cfg \
    -v $(pwd)/fxpkcs11/x64/${OPENSSL_VERSION}/lib-
fxpkcs11.so:/opt/oracle/extapi/64/hsm/futurex/${PKCS_VERSION}/libfxpkcs11.so \
    -v /home/oracle/pki:/pki \
    -p 1521:1521 \
    -p 5500:5500 \
    -e ORACLE_SID=test \
    -e ORACLE_PWD=Password123 \
    -v data:/opt/oracle/oradata \
    --name TDE \
    oracle/database:19.3.0-ee
```

   Note: The above command takes 10 to 20 minutes to complete, depending on your system's resources.You must run this command from the same directory with the extracted fxpkcs11 directory.

8. Once the Oracle Database container is running, run the following command to connect to the container's file system:

```
$ docker exec -it TDE /bin/bash
```

9. Modify the /opt/oracle/product/19c/dbhome_1/network/admin/sqlnet.ora file to the following, then save:

```
NAME.DIRECTORY_PATH= (TNSNAMES, EZCONNECT, HOSTNAME)
WALLET_LOCATION=(SOURCE=(METHOD=HSM)(METHOD_DATA=(DIRECTORY=/opt/oracle/admin/wallet)))
ENCRYPTION_WALLET_LOCATION=(SOURCE=(METHOD=HSM)(METHOD_DATA=(DIRECTORY-
Y=/opt/oracle/admin/wallet)))
WALLET_ROOT=/opt/oracle/admin/wallet
```

10. Connect to the database.

```
$ sqlplus sys/Password123@test as sysdba
```

11. Create the Master Encryption Key for TDE.

```
SQL > ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "safest";
```

12. Upon success, the following message appears:

```
System altered.
```

[8] OPENING THE WALLET/HARDWARE KEYSTORE

The security administrator must make the KMES Series 3 accessible to the database before Oracle TDE can perform any encryption or decryption. This is comparable to opening the Oracle wallet or logging in to the hardware keystore. The wallet/hardware keystore can be opened manually or automatically, but with the manual option, you must re-enable access to the KMES every time the database is restarted. Both methods are described below.

[8.1] MANUAL OPTION

Run the following command to open the hardware keystore manually, thus making the KMES accessible:

```
SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY "KMES_User_Password";
```

You can disable access with the following command.

```
SQL> ALTER SYSTEM SET ENCRYPTION WALLET CLOSE IDENTIFIED BY "KMES_User_Password";
```

Note: You must re-enable access to the KMES every time you restart the database instance, if using the manual option.

[8.2] AUTOMATIC OPTION

Note: An auto-login wallet stores the KMES credentials in an auto-login software keystore. This configuration reduces the security of the system as a whole; however, this configuration supports unmanned or automated operations, and is useful in deployments where automatic re-login to the KMES is necessary.

1. Create the **/etc/ORACLE/WALLETS/tde** directory path using the **mkdir** command below:

   ```
   $ sudo mkdir -p /etc/ORACLE/WALLETS/tde
   ```

2. Change ownership of the **/etc/ORACLE** directory to the **oracle** user.

   ```
   $ chown -R oracle:oinstall /etc/ORACLE
   ```

3. Set the **WALLET_ROOT** parameter the **WALLETS** directory created in the first step.

   ```
   SQL> ALTER SYSTEM SET WALLET_ROOT = '/etc/ORACLE/WALLETS' SCOPE=SPFILE;
   ```

4. Set the **TDE_CONFIGURATION** parameter to **FILE** for the **KEYSTORE_CONFIGURATION**.

   ```
   SQL> ALTER SYSTEM SET TDE_CONFIGURATION="KEYSTORE_CONFIGURATION=FILE" SCOPE=SPFILE;
   ```

5. Bounce the database after setting the WALLET_ROOT and TDE_CONFIGURATION parameters by shutting it down and starting it back up.

   ```
   SQL> SHUTDOWN IMMEDIATE;

   SQL> STARTUP;
   ```

6. If you have not migrated from a software keystore, create the software keystore with the hardware keystore password in the appropriate location (e.g., /etc/ORACLE/WALLETS/tde).

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY "Software_Keystore_Password";
```

Note: The *Software_Keystore_Password* value can be any password you choose.

7. Open the new software keystore with the following command:

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

8. Add the secret to the software keystore. The secret is the KMES user password and client is HSM_PASSWORD. HSM_PASSWORD is an Oracle-defined client name that represents the HSM password as a secret in the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET 'KMES_User_Password' FOR CLIENT 'HSM_PASSWORD'
IDENTIFIED BY "Software_Keystore_Password" WITH BACKUP;
```

Note: You must provide the secret and HSM_PASSWORD values within single quotes or the command will fail.

9. Create a new auto-login keystore using the password of the Oracle software wallet.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE IDENTIFIED BY "Software_
Keystore_Password";
```

10. Re-set the **TDE_CONFIGURATION** parameter to **HSM|FILE** for the **KEYSTORE_CONFIGURATION**.

```
SQL> ALTER SYSTEM SET TDE_CONFIGURATION = 'KEYSTORE_CONFIGURATION=HSM|FILE' SCOPE=SPFILE;
```

11. Bounce the database after setting the **TDE_CONFIGURATION** parameter by shutting it down and starting it back up.

```
SQL> SHUTDOWN IMMEDIATE;

SQL> STARTUP;
```

12. At this stage, the hardware security module auto-login keystore opens automatically the next time a TDE operation executes. To confirm that the auto-login wallet is working, run the following query:

```
SQL> SELECT WRL_TYPE, WRL_PARAMETER, WALLET_TYPE, STATUS FROM V$ENCRYPTION_WALLET;
```

If the auto-login wallet was configured properly, the following output appears:

```
WRL_TYPE
--------------------
WRL_PARAMETER
--------------------------------------------------------------------------------
WALLET_TYPE          STATUS
-------------------- ------------------------------
FILE
/etc/ORACLE/WALLETS/tde/
AUTOLOGIN            OPEN_NO_MASTER_KEY

HSM

HSM                  OPEN
```

APPENDIX A: XCEPTIONAL SUPPORT

In today's high-paced environment, we know you are looking for timely and effective resolutions for your mission-critical needs. That is why our Xceptional Support Team does whatever it takes to ensure you have the best experience and support possible. Every time. Guaranteed.

- 24x7x365 mission critical support
- Level 1 to level 3 support
- Extremely knowledgeable subject matter experts

At Futurex, we strive to supply you with the latest data encryption innovations as well as our best-in-class support services. Our Xceptional Support Team goes above and beyond to meet your needs and provide you with exclusive services that you cannot find anywhere else in the industry.

- Technical Services
- Onsite Training
- Virtual Training
- Customized Consulting
- Customized Software Solutions
- Secure Key Generation, Printing, and Mailing
- Remote Key Injection
- Certificate Authority Services

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

ENGINEERING CAMPUS

864 Old Boerne Road

Bulverde, Texas, USA 78163

Phone: +1 830-980-9782

+1 830-438-8782

E-mail: info@futurex.com

XCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

SOLUTIONS ARCHITECT

E-mail: solutions@futurex.com