ORACLE DATABASE 12C TRANSPARENT DATA ENCRYPTION

Integration Guide

Applicable Devices:
Vectera Plus

TABLE OF CONTENTS

[1] DOCUMENT INFORMATION

[1.1] DOCUMENT OVERVIEW

This document provides information about configuring Futurex HSMs with Oracle Database 12c Transparent Data Encryption (TDE) using Futurex PKCS #11 libraries. For additional questions related to your HSM, see the Vectera Plus user guide.

[1.2] 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.”

[1.3] GUARDIAN INTEGRATION

The Guardian Series 3 introduces mission-critical viability to core cryptographic infrastructure, including:

- Centralization of device management
- Elimination of points of failure
- Distribution of transaction loads
- Group-specific function blocking
- User-defined grouping systems

Please see the applicable guide in the Futurex Portal, which covers how to use the Guardian Series 3 to configure HSMs for PKCS #11 integrations.

[2] ORACLE DATABASE 12C TRANSPARENT DATA ENCRYPTION (TDE) INTEGRATION OVERVIEW

Integrating Oracle Database 12c Transparent Data Encryption (TDE) with the Vectera Plus 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 Vectera Plus), 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. Futurex recommends that the connection between the Futurex PKCS #11 library and the Vectera Plus be a mutually authenticated TLS connection, but server side authentication is also supported.

Note: The instructions for configuring the Futurex PKCS #11 library with Oracle Database running in a Docker container (section 10.2) only covers using TLS certificates for mutual authentication.

This guide provides 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 Vectera Plus to be used for encrypting the Oracle Table Keys. The following diagram summarizes this process:

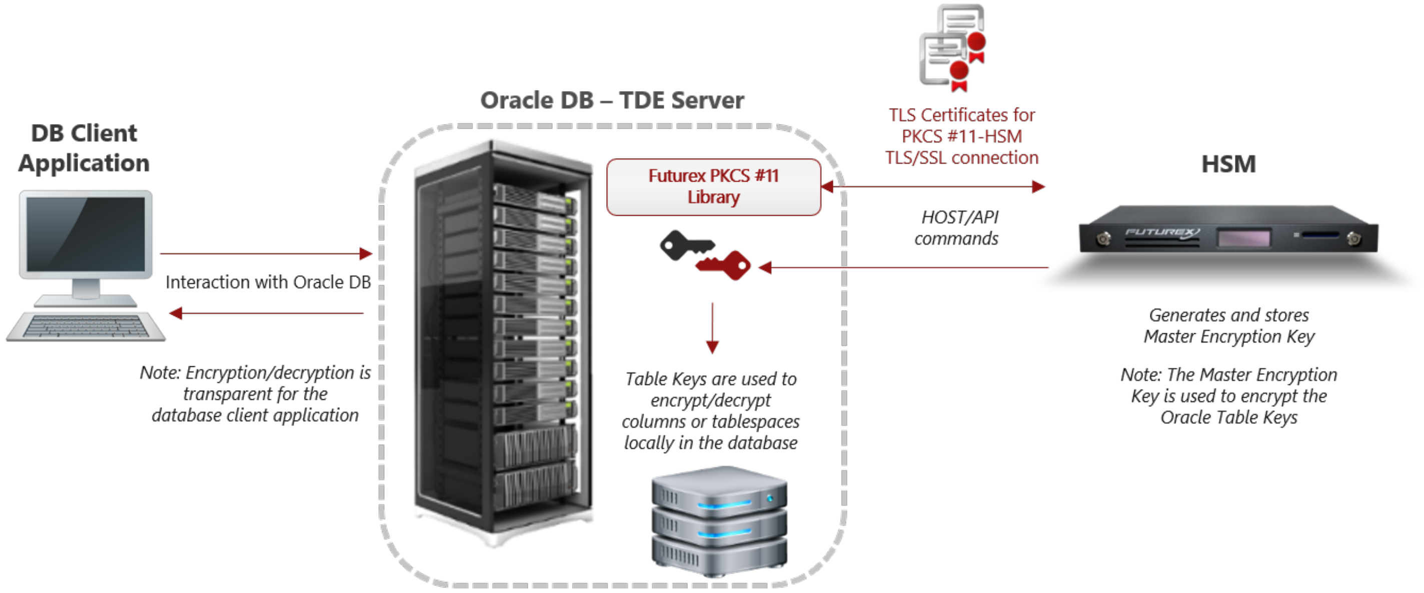

DB Client Application
Interaction with Oracle DB
Note: Encryption/decryption is transparent for the database client application
Oracle DB – TDE Server
Futurex PKCS #11 Library
Table Keys are used to encrypt/decrypt columns or tablespaces locally in the database
TLS Certificates for PKCS #11-HSM TLS/SSL connection
HOST/API commands
HSM
Generates and stores Master Encryption Key
Note: The Master Encryption Key is used to encrypt the Oracle Table Keys

FIGURE: PROCESS OF ORACLE TDE INTEGRATION

[3] PREREQUISITES

1. Linux Server (CentOS 7 for this example – GUI not required): https://www.centos.org/download/
2. OpenSSL for Linux
3. Oracle Database 12c
4. Installation Guide for Oracle Database 12c: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/ladbi/index.html
5. Oracle Guide to securing data with Oracle TDE (General concepts – how to enable TDE): https://docs.oracle.com/cd/B28359_01/network.111/b28530/asotrans.htm#g1011122
6. Oracle Guide to use HSM with TDE: https://docs.oracle.com/cd/E11882_01/network.112/e40393/asotrans.htm#ASOAG640
7. Vectera Plus running 6.7.x.x firmware and with Futurex-signed TLS certificates preloaded
8. Futurex PKCS #11 (FXPKCS11) library version 4.34 or above

Note: For the purposes of this document, it is assumed that the Linux server is running and that the user has root permissions. It is also assumed that the Oracle database is already installed, and configured to support TDE: https://docs.oracle.com/cd/B28359_01/network.111/b28530/asotrans.htm#g1011122.

This guide shows how to configure the Futurex PKCS #11 library to be used as an interface for Oracle TDE to connect to a Vectera Plus HSM wallet. More information is provided in the following link: https://docs.oracle.com/cd/E11882_01/network.112/e40393/asotrans.htm#ASOAG640.

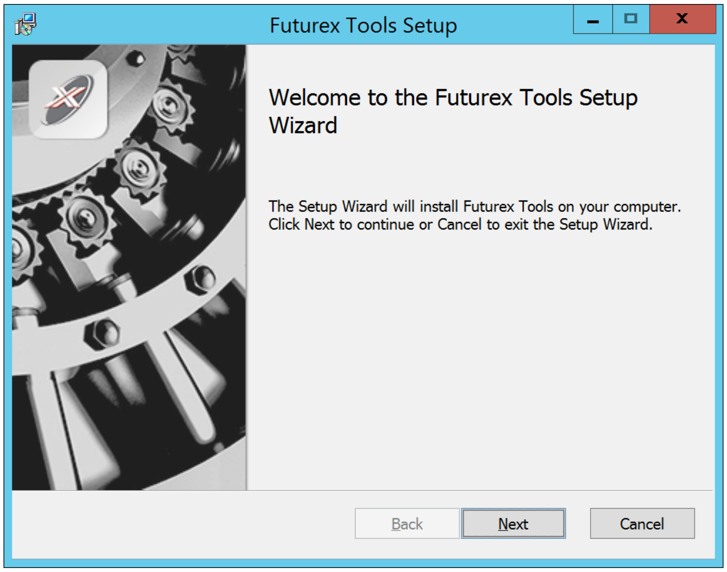
[4] INSTALL FUTUREX PKCS #11 (FXPKCS11)

In a Windows environment, the easiest way to install the **Futurex PKCS #11 (FXPKCS11)** module is with **Futurex Tools (FXTools)**. You can download FXTools from the Futurex Portal. In a Linux environment, you must download a tarball of the FXPKCS11 binaries from the Futurex Portal and then extract the tar file locally where you want the application to be installed on your system. The following sections provide step-by-step installation instructions for both of these scenarios.

Note: Install FXPKCS11 on the same computer as the application integrating with the Vectera Plus HSM.

[4.1] INSTALLING THE FXPKCS11 MODULE USING FXTOOLS IN WINDOWS

Run the Futurex Tools installer as an administrator and follow the prompts in the setup wizard to complete the installation.



Futurex Tools Setup

Welcome to the Futurex Tools Setup Wizard

The Setup Wizard will install Futurex Tools on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

FIGURE: FUTUREX TOOLS SETUP WIZARD

The Setup Wizard installs all tools on the system by default. You can override the defaults and choose not to install certain modules. The installation provides the following services:

- **Futurex Client Tools** - Command Line Interface (CLI) and associated SDK for both Java and C.
- **Futurex CNG Module**- The Microsoft Next Generation Cryptographic Library.
- **Futurex Cryptographic Service Provider (CSP)**- The legacy Microsoft cryptographic library.
- **Futurex EKM Module**- The Microsoft Enterprise Key Management library.
- **Futurex PKCS #11 Module**- The Futurex PKCS #11 library and associated tools.
- **Futurex Secure Access Client**- A client used to connect a Futurex Excrypt Touch to a local laptop through USB, which can then connect to a remote Futurex device.

If the Futurex Secure Access Client was selected, the process will also install the Futurex Excrypt Touch driver, which might start minimized or in the background.

After the installation completes, all services are installed in the C:\Program Files\Futurex\ directory. The CNG Module, CSP Module, EKM Module, and PKCS #11 Module all require configuration files, which are located in their corresponding directory with a .cfg extension. In addition, the installation registers the CNG and CSP Modules in the Windows Registry (HKEY_LOCAL_ MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider), and installs them in the C:\Windows\System32\ directory.

[4.2] INSTALLING THE FXPKCS11 MODULE IN LINUX

Extract the tarball file for your Linux distrubution in the desired working directory.

Note: To make the Futurex PKCS #11 module accessible system-wide, move it to the /usr/local/bin directory as an administrative user. If only the current user needs to use the module, then install it in $HOME/bin.

The extracted content of the tar file is a single fxpkcs11 directory. Inside the fxpkcs11 directory is the following files and directories:

- **fxpkcs11.cfg**: FXPKCS11 configuration file
- **x86/**: This folder contains the module files for 32-bit architecture
- **x64/**: This folder contains the module files for 64-bit architecture

The x86 and x64 directories each contain two subdirectories, OpenSSL-1.0.x and OpenSSL-1.1.x. These OpenSSL directories contain the following FXPKCS11 module files built with the respective OpenSSL versions:

- **configTest**: Program to test configuration and connection to the HSM
- **libfxpkcs11.so**: FXPKCS11 Library File
- **libfxpkcs11-Debug.so**: FXPKCS11 Debug Library File
- **PKCS11Manager**: Program to test connection and manage the HSM through the FXPKCS11 library

By default, the FXPKCS11 module looks for the FXPKCS11 configuration file (i.e., fxpkcs11.cfg) in the /etc directory. Alternatively, a system environment variable can be defined for the location of the FXPKCS11 configuration file. To do so permanently, open the /etc/profile file in a text editor as an administrative user, add the following line at the bottom, and save the file.

```
export FXPKCS11_CFG=/usr/local/bin/fxpkcs11/fxpkcs11.cfg
```

Note: The file location specified above must be specific to where the FXPKCS11 configuration file is saved on your system.

[5] INSTALL EXCRYPT MANAGER (IF USING WINDOWS)

Sections 4 and 5 of this integration guide cover the installation of Excrypt Manager and FXCLI. Excrypt Manager is a Windows application that provides a GUI-based method for configuring the HSM, while FXCLI provides a command-line-based method for configuring the HSM and can be installed on all platforms.

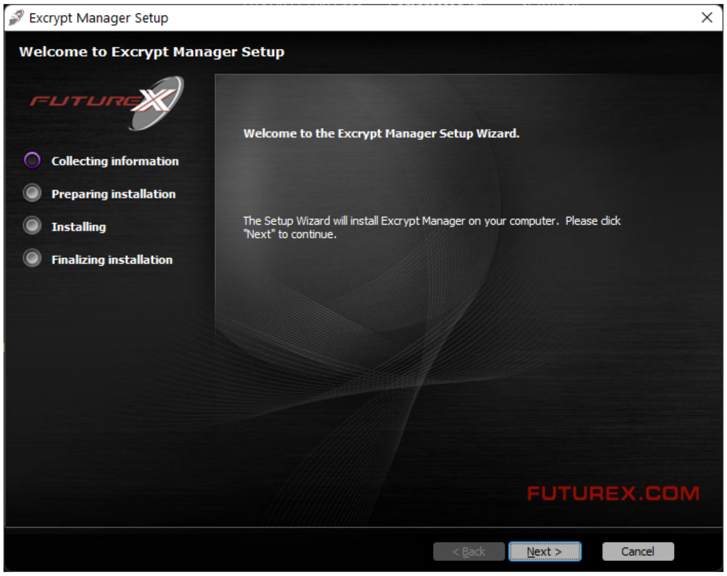
Note: If you will be configuring the Vectera Plus from a Linux computer, you can skip this section. If you will be configuring the Vectera Plus from a Windows computer, installing FXCLI in the next section is still required because FXCLI is the only method that can be used to configure TLS certificates in section 6.7.

Note: Install Excrypt Manager on the workstation you will use to configure the HSM.

Note: If you plan to use a Virtual HSM for the integration, all configurations will need to be performed using either FXCLI, the Excrypt Touch, or the Guardian Series 3.

Note: The Excrypt Manager version must be from the 4.4.x branch or later to be compatible with the HSM firmware, which must be 6.7.x.x or later.

To install Excrypt Manager, run the Excrypt Manager installer as an administrator and follow the prompts in the setup wizard to complete the installation.



Welcome to Excrypt Manager Setup

Welcome to the Excrypt Manager Setup Wizard.

The Setup Wizard will install Excrypt Manager on your computer. Please click "Next" to continue.

FIGURE: EXCRYPT MANAGER SETUP WIZARD

The installation wizard prompts you to specify where you want to install Excrypt Manager. The default location is C:\Program Files\Futurex\Excrypt Manager\. After choosing a location, select **[Install]**.

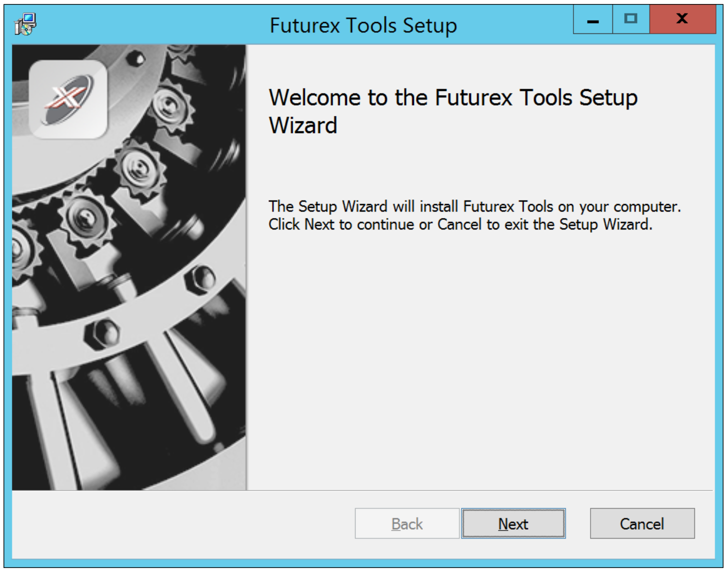
[6] INSTALL FUTUREX COMMAND LINE INTERFACE (FXCLI)

Note: Install FXCLI on the workstation you will use to configure the HSM.

[6.1] INSTALLING FXCLI IN WINDOWS

As mentioned in section 3, the FXTools installation package includes Futurex Client Tools (FXCLI). Similar to the Futurex PKCS #11 (FXPKCS11) module, the easiest way to install FXCLI on Windows is by installing FXTools. You can download FXTools from the Futurex Portal.

To install FXCLI, run the Futurex Tools installer as an administrator and follow the prompts in the setup wizard to complete the installation.



Futurex Tools Setup

Welcome to the Futurex Tools Setup Wizard

The Setup Wizard will install Futurex Tools on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

FIGURE: FUTUREX TOOLS SETUP WIZARD

The setup wizard installs all tools on the system by default. You can override the defaults and choose not to install certain modules. The installation provides the following services:

- **Futurex Client Tools**:Command Line Interface (CLI) and associated SDK for both Java and C.
- **Futurex CNG Module**:The Microsoft Next Generation Cryptographic Library.
- **Futurex Cryptographic Service Provider (CSP)**:The legacy Microsoft cryptographic library.
- **Futurex EKM Module**:The Microsoft Enterprise Key Management library.
- **Futurex PKCS #11 Module**:The Futurex PKCS #11 library and associated tools.
- **Futurex Secure Access Client**:A client used to connect a Futurex Excrypt Touch to a local laptop through USB, which can then connect to a remote Futurex device.

[6.2] INSTALLING FXCLI IN LINUX

Download FXCLI

You can download the appropriate FXCLI package files for your system from the Futurex Portal.

If the system is **64-bit**, select from the files marked **amd64**. If the system is **32-bit**, select from the files marked **i386**.

If running an OpenSSL version in the **1.0.x** branch, select from the files marked **ssl1.0**. If running an OpenSSL version in the **1.1.x** branch, select from the files marked **ssl1.1**.

Futurex offers the following features for FXCLI:

- Java Software Development Kit (**java**)
- HSM command line interface (**cli-hsm**)
- KMES command line interface (**cli-kmes**)
- Software Development Kit headers (**devel**)
- YAML parser used to parse bash output (**cli-fxparse**)

Install FXCLI

To install an rpm package, run the following command in a terminal:

```
$ sudo rpm -ivh [fxcl-xxxx.rpm]
```

To install a deb package, run the following command in a terminal:

```
$ sudo dpkg -i [fxcl-xxxx.deb]
```

Running FXCLI

To enter the HSM FXCLI prompt, run the following command in a terminal:

```
$ fxcli-hsm
```

After entering the FXCLI prompt, you can run **help** to list all of the available FXCLI commands.

[7] CONFIGURE THE VECTERA PLUS

To establish a connection between the Futurex PKCS #11 library and the Vectera Plus, perform the following configuration steps:

Note: You can complete all of the steps in this section using either Excrypt Manager or FXCLI (except for section 7.7.2, which can only be completed using FXCLI). Optionally, you can complete steps 4 through 6 using the Guardian Series 3 (Please refer to the applicable guide for configuring HSMs for PKCS #11 integrations using the Guardian Series 3).

1. Connect to the HSM through the front USB port. (**Note:** If you are using a virtual HSM for the integration, you must connect to it over the network through FXCLI, the Excrypt Touch, or the Guardian Series 3):
    a. Connecting via Excrypt Manager
    b. Connecting via FXCLI
2. Validate that the correct features are enabled on the HSM.
3. Set up the network configuration.
4. Load the Futurex FTK.
5. Configure a Transaction Processing connection and create a new Application Partition.
6. Create a new identity that has access to the newly created Application Partition.
7. Configure TLS Authentication by using one of the following options:
    a. Enable server-side authentication.
    b. Create client certificates for mutual authentication.

Each of these action items is detailed in the following subsections.

[7.1] CONNECT TO THE HSM THROUGH THE FRONT USB PORT

Note: For both Excrypt Manager and FXCLI you need to connect your laptop to the front USB port on the HSM.

Connecting through Excrypt Manager

1. Open Excrypt Manager and click **[Refresh]** in the lower right-hand side of the Connection menu. Then, select **USB Connection** and click **[Connect]**.
2. Log in with both default Admin identities.
3. You must change the default Admin passwords for both of your default Admin identities (**Admin1** and **Admin2**) to load the major keys onto the HSM. To do so via Excrypt Manager, open the **Identity Management** menu, select the first default Admin identity (**Admin1**), and select **[Change Password…]**. Enter the old password and enter the new password twice. Select **[OK]**. Perform the same steps for the second default Admin identity (**Admin2**).

Connecting through FXCLI

1. Start the FXCLI application and run the following commands:

```
$ connect usb
$ login user
```

Note: The **login** command prompts for the username and password. You must run the command twice because you must login with both default Admin identities.

2. You must change the default Admin passwords for both of your default Admin Identities in order to load the major keys onto the HSM. Use the following FXCLI commands to change the passwords for each default Admin Identity.

```
$ user change-password -u Admin1
$ user change-password -u Admin2
```

Note: The preceding **user change-password** commands prompt you to enter the old and new passwords.

[7.2] REQUIRED FEATURES IN HSM

To establish a connection between the Futurex PKCS #11 Library and the Vectera Plus, the HSM must be configured with the following features:

- **PKCS #11** > *Enabled*.
- **Command Primary Mode** > *General Purpose* (GP).

Note: For additional information about how to update features on your HSM, refer to the **"Download Feature Request File"** section of the Vectera Plus user guide.

Note: Setting the **Command Primary Mode** on the HSM to *General Purpose (GP)* enables the option to create the FTK major key in the HSM. This key is required to be able to use the Futurex PKCS #11 library to communicate with the HSM. For detailed information about how to load major keys on the HSM, refer to the Vectera Plus user guide.

[7.3] NETWORK CONFIGURATION (SETTING THE HSM IP ADDRESS)

Note: For this step you need to be logged in with an identity that has a role with permissions **Communication:Network Settings**. You can use the default Administrator role and Admin identities.

Excrypt Manager

1. Navigate to the **Configuration** menu and modify the IP configuration as required.

FXCLI

1. Run the **network interface modify** FXCLI command to set an IP for the HSM. An example is provided below to show the command syntax:

```
$ network interface modify --interface Ethernet1 --ip 10.221.0.10 --netmask 255.255.255.0 --
gateway 10.221.0.1
```

Note: At this point during the HSM configuration, consider the following:

- You can complete the remaining HSM configurations in this section using the Guardian Series 3 (see the applicable guide for configuring HSMs for PKCS #11 integrations using the Guardian Series 3), except for the final subsection, which covers creating connection certificates for mutual authentication.
- If you are performing the configuration on the HSM directly right now, but plan to add the HSM to a Guardian later, you might have to synchronize the HSM after you add it to a Device Group on the Guardian.
- If your use-case requires configuration through a CLI, then you should manage the HSMs directly.

[7.4] LOAD FUTUREX KEY (FTK)

Note: For this step you need to be logged in with an identity that has a role with permissions **Major Keys:Load**. You can use the default Administrator role and Admin identities.

The FTK wraps all keys stored on the HSM used with PKCS #11. If using multiple HSMs in a cluster, you can use the same FTK for syncing HSMs. An HSM must have an FTK before you can use it with PKCS #11.

Excrypt Manager

1. Navigate to the **Key Management** menu, then select the **Load** button for the FTK in the Major Keys section. You can load keys loaded that are XOR'd together, M-of-N fragments, or generated. If this is the first HSM in a cluster, we recommend you generate the key and save to smart cards as M-of-N fragments.

FXCLI

1. Run the following **majorkey** FXCLI commands to load an FTK into an HSM. You must generate a random FTK if this is the first HSM you are setting up. Optionally, you can also load an FTK onto smart cards simultaneously with the **-m** and **-n** flags, as shown in the following example:

   ```
   $ majorkey random --ftk -m [number_from_2_to_9] -n [number_from_2_to_9]
   ```

   If it is a second HSM you're setting up in a cluster, load the FTK from smart cards with the following command:

   ```
   $ majorkey recombine --key ftk
   ```

[7.5] CONFIGURE A TRANSACTION PROCESSING CONNECTION AND CREATE AN APPLICATION PARTITION

Note: For this step you need to be logged in with an identity that has a role with permissions **Role:Add**, **Role:Assign All Permissions**, **Role:Modify**, **Keys:All Slots**, and **Command Settings:Excrypt**. You can use the default Administrator role and Admin identities.

Note: For the purposes of this integration guide, the terms *Application Partition* and *Role* are synonymous.

[7.5.1] Configure a Transaction Processing connection

Before an application logs in to the HSM with an authenticated user, it first connects through a Transaction Processing connection to the **Transaction Processing** Application Partition. For this reason, you must take steps to harden this Application Partition. The following items need to be configured for the Transaction Processing partition:

- It should not have access to the **All Slots** permissions.
- It should not have access to any key slots.
- Only the PKCS #11 communication commands should be enabled.

Excrypt Manager

1. Navigate to the **Application Partitions** menu, select the **Transaction Processing** Application Partition, and click **[Modify...]**.
2. In the **Permissions** tab, leave the top-level **Keys** permission checked, but uncheck the **All Slots** sub permission.
3. In the **Key Slots** tab, ensure that the settings do not specify key ranges. By default, the Transaction Processing Application Partition has access to the entire range of key slots on the HSM.
4. In the **Commands** tab, make sure that only the following PKCS #11 communication commands are enabled:
    - **ECHO**: Communication Test/Retrieve Version
    - **PRMD**: Retrieve HSM restrictions
    - **RAND**: Generate random data
    - **HASH**: Retrieve device serial
    - **GPKM**: Retrieve key table information
    - **GPKS**: General purpose key settings get/change
    - **GPKR**: General purpose key settings get (read-only)

FXCLI

1. Run the following **role modify** FXCLI commands to remove all permissions and key ranges that are currently assigned to the **Transaction Processing** role and enable only the PKCS #11 communication commands:

    Note: The **Transaction Processing** role was previously referred to as the **Anonymous** role. That is why *Anonymous* is specified in the name field in the commands below.

```
$ role modify --name Anonymous --clear-perms --clear-key-ranges
```

```
$ role modify --name Anonymous --add-perm "Keys" --add-perm Excrypt:ECHO --add-perm
Excrypt:PRMD --add-perm Excrypt:RAND --add-perm Excrypt:HASH --add-perm Excrypt:GPKM --add-
perm Excrypt:GPKS --add-perm Excrypt:GPKR
```

[7.5.2] Create an Application Partition

To segregate applications on the HSM, you must create an Application Partition specifically for your use case. Application partitions are used to segment the permissions and keys on an HSM between applications. The following steps outline the process for creating and configuring a new application partition.

Excrypt Manager

1. Navigate to the **Application Partitions** menu and select **[Add...]**.
2. In the **Basic Information** tab, configure all of the fields as follows:
    a. For **Role Name**, specify any name that you would like for this new Application Partition.
    b. Set **Logins Required** to *1*.
    c. Set **Ports** to *Prod*.
    d. Configure **Connection Sources** to *Ethernet*.
    e. Leave **Managed Roles** blank because you specify the exact Permissions, Key Slots, and Commands for this Application Partition or Role to have access to.
    f. Set **Use Dual Factor** to *Never*.
    g. Leave **Upgrade Permissions** unchecked.
3. In the **Permissions** tab, select the following key permissions:
    - **Keys**
    - **Authorized** (allows for keys that require login)
    - **Import PKI** (allows trusting an external PKI. Generally not recommended, but some applications use this to allow for PKI symmetric key wrapping.)
    - **No Usage Wrap** (allows for interoperable key wrapping without defining key usage as part of the wrapped key. Use this only if you want to exchange keys with external entities or use the HSM to wrap externally used keys.)
4. In the **Key Slots** tab, we recommend you create a range of 1000 total keys that do not overlap with another Application Partition. Within the specified range, you should have ranges for both symmetric and asymmetric keys. If the application requires more keys, configure accordingly.
5. Based on application requirements, particular functions need to be enabled on the Application Partition to use the HSMs functionality. For the Oracle TDE integration, the following commands need to be enabled in the **Commands** tab.

PKCS #11 Communication Commands

- **ECHO**: Communication Test/Retrieve Version
- **HASH**: Retrieve device serial
- **GPKM**: Retrieve key table information
- **GPKS**: General purpose key settings get/change

Key Operations Commands

- **GPGS**: General purpose generate symmetric key

Data Encryption Commands

- **GPSE**: General purpose data encrypt
- **GPSD**: General purpose data decrypt

FXCLI

1. Run the following **role** FXCLI commands to create the new Application Partition and enable all required functions:

```
$ role add --name Role_Name --application --key-range (0,999) --perm "Keys:Authorized" --perm
"Keys:Import PKI" --perm "Keys:No Usage Wrap"
```

```
$ role modify --name [role_name] --clear-perms --add-perm Excrypt:ECHO --add-perm Excrypt:HASH
--add-perm Excrypt:GPKM --add-perm Excrypt:GPKS --add-perm Excrypt:GPGS --add-perm
Excrypt:GPSE --add-perm Excrypt:GPSD
```

[7.6] CREATE A NEW IDENTITY AND ASSOCIATE IT WITH THE NEWLY CREATED APPLICATION PARTITION

Note: For this step you need to be logged in with an identity that has a role with the **Identity:Add** permission. You can use the default Administrator role and Admin identities.

Excrypt Manager

1. Navigate to the **Identity Management** menu and select **[Add...]**.
2. Specify a name for the new identity and open the **Roles** drop-down menu to select the name of the previously created Application Partition. This associates the new identity with the Application Partition that you created.

FXCLI

1. Run the **identity add** FXCLI command to create a new identity and associate it with the Application Partition/Role that you created:

```
$ identity add --name Identity_Name --role Role_Name --password safest
```

You must set the name of this identity in the fxpkcs11.cfg file, in the following section:

```
#HSM crypto operator identity name
<CRYPTO-OPR>     [insert name of identity that you created]     </CRYPTO-OPR>

# Production connection
<PROD-ENABLED>    YES        </PROD-ENABLED>
<PROD-PORT>        9100        </PROD-PORT>
```

[7.7] CONFIGURE TLS AUTHENTICATION

Note: For this step you need to be logged in with an identity that has a role with permissions **Keys:All Slots**, **Management Commands:Certificates**, **Management Commands:Keys, Security:TLS Sign**, and **TLS Settings:Upload Key**. You can use the default Administrator role and Admin identities.

[7.7.1] Enable server-side authentication (option 1)

Futurex recommends mutually authenticating to the HSM using client certificates, but the Vectera Plus also supports server-side authentication. The following steps outline the process for enabling server-side authentication.

Excrypt Manager

1. Navigate to the **SSL/TLS Setup** menu. Then, select the **Excrypt Port** in the Connection Pair dropdown, check the **Allow Anonymous** box, and click **[Save]**.

FXCLI

1. Run the **tls-ports set** FXCLI command to enable server-side authentication with the **Allow Anonymous** SSL/TLS setting:

```
$ tls-ports set -p "Excrypt Port" --anon
```

[7.7.2] Create Connection Certificates for mutual authentication (option 2)

As mentioned previously, Futurex recommends mutually authenticating to the HSM using client certificates, and the system enforces mutual authentication by default. In the following example, FXCLI generates a CA which is used to sign the HSM server certificate and a client certificate. The client keys and CSR are generated using OpenSSL.

Note:

- For this example, you must connect the computer that is running FXCLI to the front USB port of the HSM.
- If you do not specify a file path for commands that create an output file, FXCLI saves the file to the current working directory.
- Using user-generated certificates requires you to load a PMK on the HSM.
- If you run **help** by itself, a full list of available commands displays. You can see all of the available options for any given command by running the command name followed by**help**.

1. Enter the FXCLI prompt by running **fxcli-hsm** in a terminal.
2. Perform the following steps to create connection certificates for mutual authentication:

```
# Connect your laptop to the HSM via the USB port on the front, then run this command.
$ connect usb
```

```
# Log in with both default Admin identities. This command will prompt for the username and
password. You will need to run this command twice.
$ login user
```

```
# Generate a TLS CA and store it in an available key slot on the HSM
$ generate --algo RSA --bits 2048 --usage mak --name TlsCaKeyPair --slot next
```

```
# Create a root certificate
$ x509 sign \
    --private-slot TlsCaKeyPair \
    --key-usage DigitalSignature --key-usage KeyCertSign \
    --ca true --pathlen 0 \
    --dn 'O=Futurex\CN=Root' \
    --out TlsCa.pem
```

```
# Generate the server keys for the HSM
$ tls-ports request --pair "Excrypt Port" --file production.csr --pki-algo RSA
```

```
# Sign the server CSR with the newly created TLS CA
$ x509 sign \
    --private-slot TlsCaKeyPair \
    --issuer TlsCa.pem \
    --csr production.csr \
    --eku Server --key-usage DigitalSignature --key-usage KeyAgreement \
    --ca false \
    --dn 'O=Futurex\CN=Production' \
    --out TlsProduction.pem
```

```
# Push the signed server PKI to the production port on the HSM
$ tls-ports set --pair "Excrypt Port" \
    --enable \
    --pki-source Generated \
    --clear-pki \
    --ca TlsCa.pem \
    --cert TlsProduction.pem \
    --no-anon
```

3. Run the following OpenSSL commands from Windows PowerShell rather than from the FXCLI program to generate client keys and CSR:

```
# Generate the client keys
$ openssl genrsa -out privatekey.pem 2048
```

```
# Generate a client CSR
$ openssl req -new -key privatekey.pem -out ClientPki.csr -days 365
```

4. Using FXCLI, sign the CSR that was just generated using OpenSSL.

```
# Sign the client CSR under the root certificate that was created
$ x509 sign  \
 --private-slot TlsCaKeyPair \
 --issuer TlsCa.pem \
 --csr ClientPki.csr \
 --eku Client --key-usage DigitalSignature --key-usage KeyAgreement \
 --dn 'O=Futurex\CN=Client' \
 --out SignedPki.pem
```

5. Run the remaining commands from Windows PowerShell:

```
# Use OpenSSL to create a PKCS #12 file that can be used to authenticate, as a client, using
the Futurex PKCS #11 library
$ openssl pkcs12 -export -inkey privatekey.pem -in SignedPki.pem -certfile TlsCa.pem -out
PKI.p12
```

[8] SETTING UP THE ORACLE ENVIRONMENT AND THE FUTUREX PKCS #11 LIBRARY

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

[8.1] SET UP ORACLE DATABASE ENVIRONMENT

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
```

[8.2] SET THE ENCRYPTION_WALLET_LOCATION

The ENCRYPTION_WALLET_LOCATION parameter specifies the location of the Oracle wallet. This parameter must be modified to specify that an HSM is to be used in place of the software wallet. Use the following steps to set the ENCRYPTION_WALLET_LOCATION parameter.

1. Navigate to the **$ORACLE_HOME/network/admin** directory.

   ```
   $ env | grep ORACLE_HOME
   ORACLE_HOME=/home/oracle/app/oracle/product/12.2.0/dbhome_1

   $ cd /home/oracle/app/oracle/product/12.2.0/dbhome_1/network/admin
   ```

2. Open the **sqlnet.ora** file in a text editor. If the ENCRYPTION_WALLET_LOCATION parameter is already present, change the METHOD value to HSM. If the ENCRYPTION_WALLET_LOCATION parameter is not present, add the parameter as follows:

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

   Note: If a DIRECTORY value is present in the ENCRYPTION_WALLET_LOCATION parameter, make sure not to delete it. Although the Vectera Plus does not require a DIRECTORY value, the value is used to locate the old software wallet when migrating to HSM-based Transparent Data Encryption. Also, the DIRECTORY value might be required by tools such as the Recovery Manager (RMAN) to locate the software wallet.

3. Save and close the file.

[8.3] COPY THE FUTUREX PKCS #11 LIBRARY TO THE CORRECT 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.

[9] 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 HSM. It enables the user to modify certain configurations and set connection details. This section covers the **<HSM>** 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.

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

    # HSM crypto operator user name
    <CRYPTO-OPR>            [identity_name]            </CRYPTO-OPR>
    # Automatically login on session open
    #<CRYPTO-OPR-PASS>       [identity_password]           </CRYPTO-OPR-PASS>

    # Connection information
    <ADDRESS>               10.0.8.30     </ADDRESS>
    <PROD-PORT>             9100                    </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>
</HSM>
```

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

In the **<CRYPTO-OPR>** field, specify the name of the identity you created for the Application Partition.

The **<CRYPTO-OPR-PASS>** field allows you to specify the password of the identity configured in the **<CRYPTO-OPR>** field. This define can be used to log the application into the HSM automatically if required, but for this integration, it should be commented out as shown above.

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

In the **<PROD-PORT>** field, specify port *9100*, which is the TLS Production port on the Vectera Plus.

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

The value that needs to be set in the **<PROD-TLS-ANONYMOUS>** field depends on whether server-side authentication was configured, or mutual authentication with TLS certificates. If using server-side authentication, set the value to *YES*. If using TLS certificates for mutual authentication, set the value to *NO*.

If using TLS certificates for mutual authentication, specify the location of the TLS certificates and private key in the corresponding **<PROD-TLS-CA>**, **<PROD-TLS-CERT>**, and **<PROD-TLS-KEY>** fields. If using server-side authentication, these lines can be commented out.

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.

[10] GENERATE A TDE MASTER ENCRYPTION KEY ON THE VECTERA PLUS

To configure Oracle Database 12c TDE with an HSM, we recommend that you refer to the Oracle knowledge base article below:

- Oracle Database 12c: https://docs.oracle.com/database/121/ASOAG/configuring-transparent-data-encryption.htm

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 Vectera Plus, 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.

[10.1] STANDARD IMPLEMENTATION

1. Set the Oracle environment using the following commands:

```
$ su oracle

$ cd ~

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

2. Connect to the database:

```
$ sqlplus / as sysdba
```

3. Start the Oracle instance:

```
SQL> startup
```

4. Open the hardware keystore using the password of the identity created on the Vectera Plus:

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

5. Create the TDE Master Encryption Key using the password of the identity that was created on the Vectera Plus:

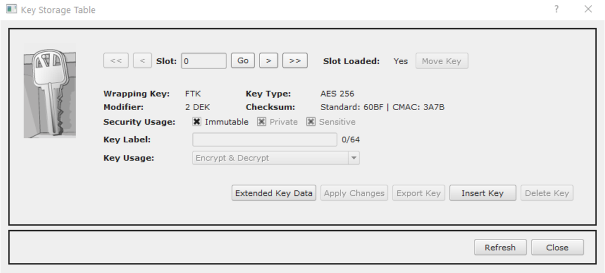
```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "Identity_Password";
```

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

6. Upon success, the following will be returned.

```
System altered.
```

Generated keys in the Vectera Plus will look like the following image:



Slot: 0 Slot Loaded: Yes

Wrapping Key: FTK Key Type: AES 256

Modifier: 2 DEK Checksum: Standard: 60BF | CMAC: 3A7B

Security Usage: ☒ Immutable ☒ Private ☒ Sensitive

Key Label: 0/64

Key Usage: Encrypt & Decrypt

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.

[10.2] DOCKER CONTAINER IMPLEMENTATION

NOTE: The steps outlined in this section are specific to how the **Futurex PKCS #11 (FXPKCS11) library** must be configured 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, navigate to the location where the Oracle Database private key (`tls_skey.pem`) file is saved.
2. Open a terminal and 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), the container is based on Oracle Linux 7, which is OpenSSL 1.0-based. In that case you would need to run the following command:

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

   If Oracle Database is running in a container that is OpenSSL 1.1-based, you would need to 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 OpenSSL 1.0-based, download the `fxpkcs11-redhat-4.xx-xxxx.tar` file. If your container is OpenSSL 1.1-based, 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. Open the FXPKCS11 configuration file (i.e., fxpkcs11.cfg) in a text editor and modify the connection details in the **<HSM>** section to allow the FXPKCS11 library to connect to the Vectera Plus. The following is a

configuration example (note that the full fxpkcs11.cfg file is not included).

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

    # HSM crypto operator user name
    <CRYPTO-OPR>            [identity_name]             </CRYPTO-OPR>
    # Automatically login on session open
    #<CRYPTO-OPR-PASS>       [identity_password]        </CRYPTO-OPR-PASS>

    # Connection information
    <ADDRESS>               10.0.8.30     </ADDRESS>
    <PROD-PORT>             9100                    </PROD-PORT>
    <PROD-TLS-ENABLED>      YES                     </PROD-TLS-ENABLED>
    <PROD-TLS-ANONYMOUS>    NO                      </PROD-TLS-ANONYMOUS>
#    <PROD-TLS-CA>           /home/user/tls/root.pem        </PROD-TLS-CA>
#    <PROD-TLS-CA>           /home/user/tls/sub1.pem     </PROD-TLS-CA>
#    <PROD-TLS-CA>           /home/user/tls/sub2.pem     </PROD-TLS-CA>
    <PROD-TLS-KEY>          /home/user/tls/PKI.p12       </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>
</HSM>
```

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

The **<CRYPTO-OPR>** field specifies the name of the identity you created for the Application Partition.

The **<CRYPTO-OPR-PASS>** field specifies the password of the identity configured in the **<CRYPTO-OPR>** field. This define can be used to log the application into the HSM automatically if required, but for this integration, it should be commented out as shown above.

The **<ADDRESS>** field specifies the IP address of the HSM that the FXPKCS11 library should connect to.

The **<PROD-PORT>** field specifies the port number of the HSM that the FXPKCS11 library should connect to.

The **<PROD-TLS-ANONYMOUS>** field defines whether the FXPKCS11 library authenticates to the server.

The **<PROD-TLS-KEY>** field defines the location of the client private key. Supported formats for the TLS private key are PKCS #1 clear private keys, PKCS #8 encrypted private keys, or a PKCS #12 file that contains the private key and certificates encrypted under the password specified in the **<PROD-TLS-KEY-PASS>** field.

Because a PKCS #12 file is defined in the **<PROD-TLS-KEY>** field in this example, the signed client cert does not need to be defined with the **<PROD-TLS-CERT>** tag, nor do the CA cert/s need to be defined with one or more instances of the **<PROD-TLS-CA>** tag.

If you use Guardian to manage HSMs in a cluster, define the **<FX-LOAD-BALANCE>**field as *YES*. Otherwise, set it to *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 Vectera Plus.

   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 Vectera Plus are not stored in the /home/oracle/pki directory on your system, modify the third -v flag in your command to reflect.

```
$ 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 will take 10 to 20 minutes to complete, depending on your system's resources.

8. Once the Oracle Database container is up and 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 output is returned:

```
System altered.
```

[11] OPENING THE WALLET/HARDWARE KEYSTORE

The security administrator must make the Vectera Plus accessible to the database before the Oracle database can perform any encryption or decryption. This 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 HSM every time the database is restarted. Both methods are described below.

[11.1] MANUAL OPTION

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

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

You can disable access with the following command:

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

Note: You must re-enable access to the HSM every time you restart the database instance with the manual option.

[11.2] AUTOMATIC OPTION

Note: An auto-login wallet stores the hardware security module 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 of the hardware security module is necessary.

1. If the hardware keystore is open, close it with the following command:

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

2. If you have not migrated from a software keystore, create the software keystore with the hardware keystore password in the appropriate location (for example, /etc/ORACLE/WALLETS/orcl).

   ```
   SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/ORACLE/WALLETS/orcl'
   IDENTIFIED BY "Software_Keystore_Password";
   ```

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

3. If you have migrated and are using an auto-login software keystore in a specific location (for example, /etc/ORACLE/WALLETS/HSM), create the software password keystore with the hardware keystore password from the auto-login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/ORACLE/WALLETS/orcl' IDENTIFIED BY "Software_Keystore_Password";

SQL> ADMINISTER KEY MANAGEMENT
MERGE KEYSTORE '/etc/ORACLE/WALLETS/HSM' -- Example keystore path
INTO EXISTING KEYSTORE '/etc/ORACLE/WALLETS/HSM' -- Example keystore location
IDENTIFIED BY "Existing_Keystore_Password" WITH BACKUP;
```

The location of the keystore for the **ADMINISTER KEY MANAGEMENT** merge statement does not need to be the location of the keystore in use.

4. Reconfigure the sqlnet.ora file and add the keystore location of the software keystore created in step 2 to the **DIRECTORY** setting of the **ENCRYPTION_WALLET_LOCATION** setting.

   ```
   ENCRYPTION_WALLET_LOCATION=
    (SOURCE=(METHOD=FILE)(METHOD_DATA=
      (DIRECTORY=/etc/ORACLE/WALLETS/orcl)))
   ```

5. To make the change take effect, either re-connect to the database, or log out and then log back in again.
6. Open the software keystore.

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

   Note: The *Software_Keystore_Password* value needs to match the value set in step 2.

7. Add or update the secret in the software keystore.

   The secret is the hardware security module password and the 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 "HSM_Identity_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 (or re-create) the auto-login keystore.

   ```
   SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE
   FROM KEYSTORE '/etc/ORACLE/WALLETS/orcl/hsm' -- Keystore location
   IDENTIFIED BY "Software_Keystore_Password";
   ```

10. Update the sqlnet.ora file to use the hardware security module location.

    ```
    ENCRYPTION_WALLET_LOCATION=
      (SOURCE=(METHOD=HSM)(METHOD_DATA=
        (DIRECTORY=/etc/ORACLE/WALLETS/orcl)))
    ```

11. At this stage, the hardware security module auto-login keystore will open automatically the next time that a TDE operation executes. To confirm that auto-login wallet is working, reboot the database then re-connect and run the following query:

    ```
    SQL> select status from v$encryption_wallet where wrl_type='HSM';
    ```

If auto-login wallet was configured properly you will see the following output:

```
STATUS
------------------------------
OPEN
```

[12] APPENDICES

APPENDIX A: XCEPTIONAL SUPPORT

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