HASHICORP VAULT - MANAGED KEYS

Integration Guide

Applicable Devices:

Vectera Plus

TABLE OF CONTENTS

[1] DOCUMENT INFORMATION

[1.1] DOCUMENT OVERVIEW

The purpose of this document is to provide information regarding the configuration of the Futurex Vectera Plus HSM with HashiCorp Vault's Managed Keys feature using Futurex PKCS #11 libraries. For additional questions related to your HSM, see the relevant user guide.

[1.2] APPLICATION DESCRIPTION

From the HashiCorp Vault documentation website: "Within certain environments, customers want to leverage key management systems external to Vault, when handling, storing, and interacting with private key material, or are required to do so by standards requirements.

To satisfy these requirements, Vault has a centralized abstraction called *Managed Keys* that different secrets engines can plug into, allowing them to delegate these operations to a trusted external KMS.

Minimally, a managed key consists of a named managed key entry managed by the sys/managed-key API. Besides a name, there are backend specific configurations to access the key in question.

For PKCS #11 (HSM) backed managed keys, the managed key configuration must reference a kms library stanza which points to a PKCS #11 access library on the host machine.

Note that a configured, named managed key corresponds to a single key within a backend. More than one managed key can be configured targeting a single backend by creating multiple managed keys with the API."

[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] PREREQUISITES

Supported Hardware:

- Vectera Plus, 6.7.x.x and above

Supported Operating Systems:

- Windows 7 and above
- Linux

Other:

- OpenSSL
- Vault 1.10 Enterprise HSM binary (can be downloaded from https://releases.hashicorp.com/vault/)

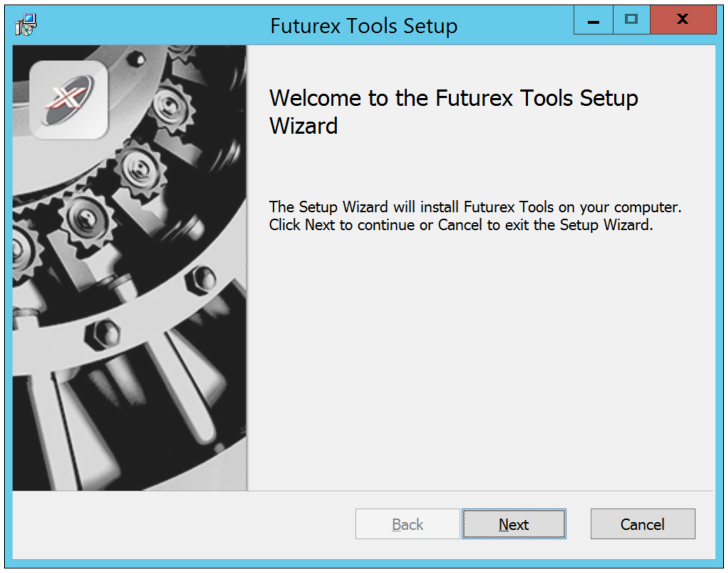
[3] INSTALL FUTUREX PKCS #11 (FXPKCS11)

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.

[3.1] 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.



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.

[3.2] 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.

[4] 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.

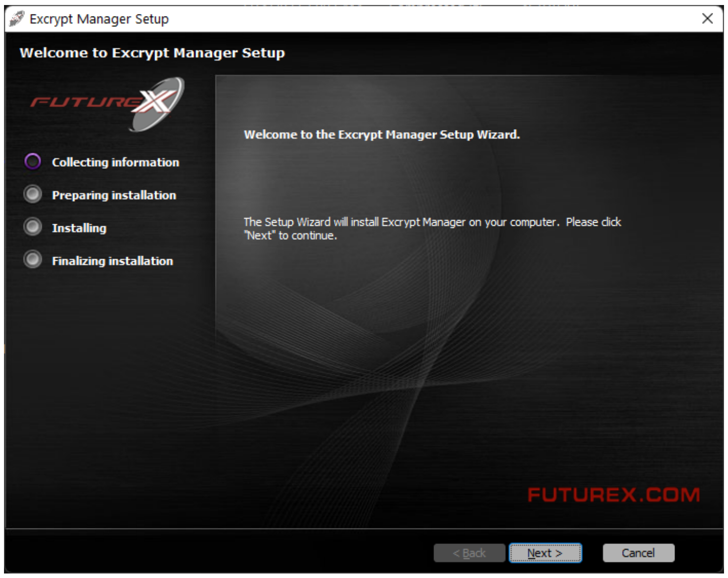
Note: If you will be configuring the Vectera Plus from a Linux computer, you can skip this section. If you will be configuring the Vectera Plus from a Windows computer, installing FXCLI in the next section is still required because FXCLI is the only method that can be used to configure TLS certificates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

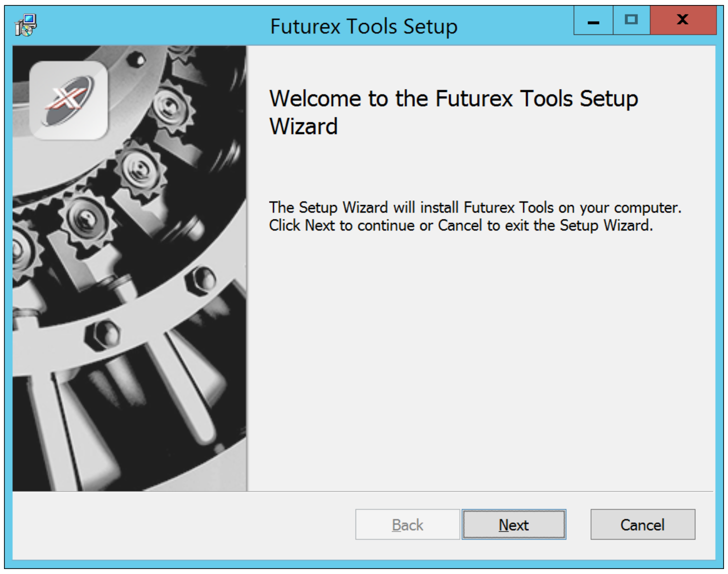
[5] INSTALL FUTUREX COMMAND LINE INTERFACE (FXCLI)

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

[5.1] 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.



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.

[5.2] 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.

[6] 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 6.7.2, 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.
8. Enable the EDSVWU multi-usage combination for asymmetric keys

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

[6.1] 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.

[6.2] 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.

[6.3] 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.

[6.4] 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
   ```

[6.5] 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.

[6.5.1] 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
   ```

[6.5.2] 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. The commands that Vault Managed Keys requires are listed on the next page. These can be enabled in the **Commands** tab.

PKCS #11 Communication Commands

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

Key Operations Commands

- **GRSA**: Generate RSA Private and Public Key
- **LRSA**: Load key into RSA Key Table

Data Encryption Commands

- **GPSR**: General purpose RSA encrypt/decrypt or sign/verify with recovery

Miscellaneous Commands

- **TIME**: Get/set the HSM internal clock.

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:GPKM
--add-perm Excrypt:GPKS --add-perm Excrypt:HASH --add-perm Excrypt:GRSA --add-perm
Excrypt:LRSA --add-perm Excrypt:GPSR --add-perm Excrypt:TIME
```

[6.6] 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>
```

[6.7] 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.

[6.7.1] 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
```

[6.7.2] 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
```

[6.8] ENABLE THE EDSVWU MULTI-USAGE COMBINATION FOR ASYMMETRIC KEYS

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

Vault Managed Keys requires asymmetric keys with multiple usages, which can be configured, but is not enabled by default on the Vectera Plus. The specific multi-usage combination that is needed is EDSVWU.

Excrypt Manager

1. Navigate to the *Extended Options* menu. In the Usage section, select **Asymmetric Authorize** in the dropdown , then click **[Add]**.
2. Select the EDSVWU usage combination and click **[Ok]**.
3. Click the **[Save]** button on the bottom-right-hand side of the window to save the changes.

FXCLI

1. Run the **multi-usage add** FXCLI command below to add the EDSVWU multi-usage combination for asymmetric keys for authorized users:

```
$ multi-usage add --asymmetric --auth -edsvwu
```

[7] EDIT THE FUTUREX PKCS #11 CONFIGURATION FILE

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/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*.

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 can be used to log the application into the HSM automatically, if required.

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

In the **<PROD-PORT>** field, specify the port number of the HSM that the FXPKCS11 library should connect to.

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

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*.

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.

[7.1] SPECIAL COMPATIBILITY MODE CONFIGURATION REQUIRED FOR THIS INTEGRATION

This integration requires two special defines in the **<CONFIG>** section of the fxpkcs11.cfg file.

```
<FORCED-LABEL-USAGE> hsm_demo = ENCRYPT | DECRYPT </FORCED-LABEL-USAGE>
<FORCED-LABEL-USAGE> hsm_hmac_demo = SIGN | VERIFY </FORCED-LABEL-USAGE>
```

These defines force specific usages for two of the keys that Vault creates on the HSM, based on the key labels that are specified.

Note: The **hsm_demo** and **hsm_hmac_demo** key labels correspond with what is defined for the **key_label** and **hmac_key_label** values in the vault.hcl file (covered in section 8.4).

[8] STEPS TO CONFIGURE THE FUTUREX PKCS #11 LIBRARY WITH HASHICORP VAULT

Note: Vault's Hardware Security Module (HSM) Managed Keys feature requires Vault Enterprise with the Advanced Data Protection Module.

[8.1] DOWNLOAD VAULT

Precompiled Vault binaries are available for download at https://releases.hashicorp.com/vault/ and Vault Enterprise binaries are available for download by following the instructions made available to HashiCorp Vault customers.

This integration requires the Vault 1.10 Enterprise HSM binary. It is available at this link to use for testing: https://releases.hashicorp.com/vault/1.10.0+ent.hsm/

[8.2] INSTALL VAULT

1. Unzip the downloaded package and move the *vault* binary to */usr/local/bin/*.

```
$ unzip vault_${VAULT_VERSION}+ent.hsm_linux_amd64.zip
```

2. Set the owner of the Vault binary.

```
$ sudo chown root:root vault
```

3. Check that vault is available on the system path.

```
$ sudo mv vault /usr/local/bin/
```

4. Verify the Vault version.

```
$ vault --version
```

5. The **vault** command features opt-in autocompletion for flags, subcommands, and arguments (where supported).

   Install autocompletion using the following command.

```
$ vault -autocomplete-install
```

6. Enable autocompletion.

```
$ complete -C /usr/local/bin/vault vault
```

7. Give Vault the ability to use the mlock syscall without running the process as root. The mlock syscall prevents memory from being swapped to disk.

```
$ sudo setcap cap_ipc_lock=+ep /usr/local/bin/vault
```

8. Create a unique, non-privileged system user to run Vault.

```
$ sudo useradd --system --home /etc/vault.d --shell /bin/bash vault
```

[8.3] CONFIGURE SYSTEMD

Systemd uses documented sane defaults so only non-default values must be set in the configuration file.

1. Create a Vault service file at /etc/systemd/system/vault.service.

```
$ sudo touch /etc/systemd/system/vault.service
```

2. Add the below configuration to the Vault service file:

```
[Unit]
Description="HashiCorp Vault - A tool for managing secrets"
Documentation=https://www.vaultproject.io/docs/
Requires=network-online.target
After=network-online.target
ConditionFileNotEmpty=/etc/vault.d/vault.hcl
StartLimitIntervalSec=60
StartLimitBurst=3

[Service]
User=vault
Group=vault
ProtectSystem=full
ProtectHome=read-only
PrivateTmp=yes
PrivateDevices=yes
SecureBits=keep-caps
AmbientCapabilities=CAP_IPC_LOCK
Capabilities=CAP_IPC_LOCK+ep
CapabilityBoundingSet=CAP_SYSLOG CAP_IPC_LOCK
NoNewPrivileges=yes
ExecStart=/usr/local/bin/vault server -config=/etc/vault.d/vault.hcl
ExecReload=/bin/kill --signal HUP $MAINPID
KillMode=process
KillSignal=SIGINT
Restart=on-failure
RestartSec=5
TimeoutStopSec=30
StartLimitInterval=60
StartLimitIntervalSec=60
StartLimitBurst=3
LimitNOFILE=65536
LimitMEMLOCK=infinity

[Install]
WantedBy=multi-user.target
```

[8.4] CONFIGURE VAULT

Vault uses documented sane defaults so only non-default values must be set in the configuration file.

1. Create */etc/vault.d* directory.

```
$ sudo mkdir --parents /etc/vault.d
```

2. Create a Vault configuration file, *vault.hcl*.

```
$ sudo touch /etc/vault.d/vault.hcl
```

3. Set the ownership of the */etc/vault.d* directory.

```
$ sudo chown --recursive vault:vault /etc/vault.d
```

4. Set the required file permissions.

```
$ sudo chmod 640 /etc/vault.d/vault.hcl
```

Configure Managed Keys

The kms_library stanza isolates platform specific configuration for managed keys. It defines logical names that are referenced within an API configuration keeping cluster and node specific details separated along with deployment concerns for each.

To integrate the Vault Enterprise server with an HSM for supporting the Managed Keys feature, the configuration file must define the kms_library stanza providing necessary connection information. Optionally, the pkcs11 seal stanza can also be included in the configuration file if you want to use the HSM with PKCS11 as the seal wrapping mechanism.

Example: *vault.hcl*

```
# Provide your Futurex HSM connection information
kms_library "pkcs11" {
  name="hsm1"
  library = "/usr/local/bin/fxpkcs11/libfxpkcs11-Debug.so"
}

seal "pkcs11" {
    lib = "/usr/local/bin/fxpkcs11/libfxpkcs11-Debug.so"
    slot = 0
    key_label="hsm_demo"
    hmac_key_label="hsm_hmac_demo"
    generate_key ="true"
    mechanism=0x1
}

storage "file" {
  path    = "/tmp/vault"
}

listener "tcp" {
  address     = "0.0.0.0:8200"
  tls_disable = "true"
}

disable_mlock = true
license_path = "/usr/local/bin/License.txt"

api_addr = "http://127.0.0.1:8200"
cluster_addr = "https://127.0.0.1:8201"
ui = true
```

Note: For the purpose of this guide, the storage backend is set to the local file system (*/tmp/vault*) to make the verification step easy.

Note: Save your Vault license to a file on disk. In the config file above this is specified as "License.txt".

The example configuration defines the following in its **kms_library** stanza.

Note: Multiple kms_library stanza's can be defined with the only limitation that the value for the name key needs to be a unique value across all the stanza definitions in a case-insensitive manner.

Parameter	Description
name	The logical name to be referenced by a managed key
library	The path to the PKCS #11 library shared object file.

The example configuration defines the following in its **seal stanza**:

Parameter	Description
lib	Path to the PKCS #11 library shared object file.
slot	The slot number to use (this should be set to "0" because "0" is the slot that is set by default in the FXPKCS11 configuration file).
key_label	Defines the label of the key to use.
hmac_key_label	Defines the label of the key to use for HMACing.
generate_key	If no existing key with the label specified by key_label can be found at Vault initialization time, Vault generates a key.

Note: For this integration, the **generate_key** parameter needs to be set to "true" so that Vault will automatically create the encryption keys that it uses for the Seal Wrap functionality on the HSM. The values set for the **key_label** and **hmac_key_label** parameters correspond with the special key label defines that must be set in the **<CONFIG>** section of the *fxpkcs11.cfg* file (covered in section 7.2).

For the full list of configuration parameters, please refer to the Vault documentation here.

[8.5] START THE VAULT SERVER

1. Log in with the **vault** user.
2. Set the PKCS #11 PIN for login with the following command (this is the password of the identity created on the HSM and defined in the FXPKCS11 configuration file).

```
$ export VAULT_HSM_PIN='safest'
```

   Note: The PKCS #11 PIN can also be set in the Vault configuration file (i.e., *vault.hcl*) with the **pin** parameter, but this is not recommended in a production setting. Best practice is to specify the pin with the VAULT_HSM_PIN environment variable, as shown here. This prevents the password from being exposed if the configuration file is compromised or stored in an unsecure location. If set via the environment variable, Vault will obfuscate the environment variable after reading it. The one caveat is that the VAULT_HSM_PIN environment variable will need to be re-set if Vault is restarted.

3. Start the Vault server.

```
$ vault server -config=/etc/vault.d/vault.hcl
```

If the above command is successful, something similar to the following output is expected:

```
==> Vault server configuration:

             Api Address: http://127.0.0.1:8200
                     Cgo: enabled
         Cluster Address: https://127.0.0.1:8201
              Go Version: go1.17.7
              Listener 1: tcp (addr: "0.0.0.0:8200", cluster address: "0.0.0.0:8201", max_
request_duration: "1m30s", max_request_size: "33554432", tls: "disabled")
               Log Level: info
                   Mlock: supported: true, enabled: false
           Recovery Mode: false
                 Storage: file
                 Version: Vault v1.10.0+ent.hsm
             Version Sha: d71d7710888891761ce43ec4e5f9d9fdeff31d8e

==> Vault server started! Log data will stream in below:
```

4. Open a new terminal window and leave the terminal running where the Vault server was started.

[9] TESTING PKI OPERATIONS

In this section, we will run through several PKI operations to demonstrate how to create Root and Intermediate CAs, which can then issue leaf certificates under them.

[9.1] INITIALIZE VAULT

Before performing the PKI operations listed above, you must initialize, unseal (if required), and login to Vault.

1. In a different terminal window from where Vault is running, set the VAULT_ADDR and PIN environment variables.

```
$ export VAULT_ADDR='http://127.0.0.1:8200'

$ export PIN='safest'
```

2. Check the Vault status.

```
$ vault status
```

The output should be similar to the following:

```
Key                        Value
---                        -----
Recovery Seal Type         pkcs11
Initialized                false
Sealed                     true
Total Recovery Shares      0
Threshold                  0
Unseal Progress            0/0
Unseal Nonce               n/a
Version                    n/a
HA Enabled                 false
```

3. Initialize Vault.

```
$ vault operator init -recovery-shares=1 -recovery-threshold=1
```

Note: Using "1" for both the recovery shares and the recovery threshold is not recommended in production.

The output should be similar to the following:

```
Recovery Key 1: qK4pHBY46Zxg2nt/cMgeLGh01Kh9SQ1ChOIDHPe/kmg=

Initial Root Token: hvs.iYjhzPiwz00bpqX6rmzSe7yj

Success! Vault is initialized

Recovery key initialized with 1 key shares and a key threshold of 1. Please
securely distribute the key shares printed above.
```

4. If HSM auto unseal is not configured, you must unseal Vault manually.

```
$ vault operator unseal <Recovery Key 1 provided from above>
```

5. Login to Vault.

```
$ vault login <Initial Root Token provided from above>
```

[9.2] GENERATE MANAGED KEYS ON THE HSM FOR THE ROOT AND INTERMEDIATE CA

1. Generate a managed key on the HSM for the Root CA.

```
$ vault write /sys/managed-keys/pkcs11/hsm-key-root library=hsm1 token_label=Futurex pin=$PIN
key_label="hsm-key-root" allow_generate_key=true allow_store_key=true mechanism=0x0001 key_
bits=2048 any_mount=false
```

Note: The value specified in the **library** field in the command above must match the value set in the **name** field of the **kms_library** stanza in the Vault configuration file (shown below). The value specified in the **token_label** field above always needs to be **Futurex**.

```
# Provide your Futurex HSM connection information
kms_library "pkcs11" {
  name="hsm1"
  library = "/usr/local/bin/fxpkcs11/libfxpkcs11-Debug.so"
}
```

2. Generate a managed key on the HSM for the Intermediate CA.

```
$ vault write /sys/managed-keys/pkcs11/hsm-key-int library=hsm1 token_label=Futurex pin=$PIN
key_label="hsm-key-int" allow_generate_key=true allow_store_key=true mechanism=0x0001 key_bits-
s=2048 any_mount=false
```

3. Verify that the key configurations has been written to Vault.

```
$ vault list /sys/managed-keys/pkcs11
```

4. Verify that the key configurations are valid by test signing some data.

```
$ vault write -f /sys/managed-keys/pkcs11/hsm-key-root/test/sign

$ vault write -f /sys/managed-keys/pkcs11/hsm-key-int/test/sign
```

[9.3] ENABLE THE THE PKI SECRETS ENGINE FOR THE ROOT AND INTERMEDIATE CA

1. Enable the PKI secrets engine for the Root CA.

```
$ vault secrets enable -path=pki -allowed-managed-keys=hsm-key-root pki
```

2. Enable the PKI secrets engine for the Intermediate CA.

```
$ vault secrets enable -path=pki_int -allowed-managed-keys=hsm-key-int pki
```

[9.4] CREATE A ROOT CA CERTIFICATE WITH THE CORRESPONDING MANAGED KEY THAT WAS GENERATED AND STORED ON THE HSM

1. Create a Root CA certificate with its corresponding managed key and output it to a file.

```
$ vault write -field=certificate pki/root/generate/kms managed_key_name=hsm-key-root common_
name=example.com ttl=8760h > /tmp/CA_cert.crt
```

2. Verify that the certificate looks correct.

```
$ cat /tmp/CA_cert.crt
```

[9.5] CREATE A CERTIFICATE SIGNING REQUEST (CSR) FOR THE INTERMEDIATE CA WITH THE MANAGED KEY THAT WAS GENERATED AND STORED ON THE HSM

1. Create an Intermediate CA certificate with its corresponding managed key and output it to a file.

   Note: The command below requires the **jq** package to be installed on your system. jq is used to process JSON output.

```
$ vault write -format=json pki_int/intermediate/generate/kms managed_key_name=hsm-key-int com-
mon_name="example.com" | jq -r '.data.csr' > /tmp/pki_intermediate.csr
```

2. Verify that the certificate looks correct.

```
$ cat /tmp/pki_intermediate.csr
```

[9.6] SIGN THE INTERMEDIATE CA CERTIFICATE WITH THE MANAGED ROOT CA

1. Sign the Intermediate CA certificate with the managed Root CA and output it to a file.

   Note: The command below requires the **jq** package to be installed on your system. jq is used to process JSON output.

```
$ vault write -format=json pki/root/sign-intermediate csr=@/tmp/pki_intermediate.csr form-
at=pem_bundle ttl="43800h" | jq -r '.data.certificate' > /tmp/intermediate.cert.pem
```

2. Write the signed Intermediate CA certificate to Vault.

```
$ vault write pki_int/intermediate/set-signed certificate=@/tmp/intermediate.cert.pem
```

[9.7] ISSUE A LEAF CERTIFICATE FROM THE INTERMEDIATE CA

1. Create a new role.

```
$ vault write pki_int/roles/example-dot-com allowed_domains="example.com" allow_sub-
domains=true max_ttl="720h"
```

2. Issue a leaf certificate.

```
$ vault write -format=json pki_int/issue/example-dot-com common_name="test.example.com" ttl-
l="24h"
```

APPENDIX A: CONFIGURING VAULT TO USE A KEY THAT WAS MANUALLY GENERATED ON THE HSM

To demonstrate this process, we will use Futurex Command Line Interface (FXCLI) to generate a new key on the Vectera Plus and assign it a PKCS11 label which Vault can reference when creating a new managed key.

[9.8] CONNECT AND LOG IN TO THE HSM VIA FXCLI

1. Run the FXCLI application.
2. Configure TLS certificates for communication between FXCLI and the HSM using the `tls` set of commands.

   NOTE: Run `tls help` to access syntax documentation.
3. Connect to the HSM using the following command:

```
$ connect tcp --connect hsm_ip:9009
```

4. Log in to the HSM with the default "Admin1" and "Admin2" identities by running the following command twice (each time it will prompt for username and password):

```
$ login user
```

[9.9] CREATE A NEW KEY PAIR ON THE VECTERA PLUS

1. Create a new key pair in the next available key slot on the HSM:

```
$ generate --algo RSA --bits 2048 --name VaultManualKey --slot next --tpk-slot next --usage
encrypt,decrypt,sign,verify,wrap,unwrap
```

2. Confirm which key slot the private key was added to:

```
$ keytable list
```

3. Assign a PKCS11 label to the key (Vault needs this external data field to be set so that it can find the key):

   Note: The number that you specify in the **slot** flag needs to match the slot number of the private key determined in the previous step. The PKCS11 label value should match the name set for the key pair in the **generate** command.

```
$ keytable extdata --slot 0 --p11-attr label --p11-value VaultManualKey
```

[9.10] CREATE A MANAGED KEY IN VAULT BY REFERENCING THE PKCS11 LABEL OF THE KEY THAT WAS MANUALLY GENERATED ON THE VECTERA PLUS USING FXCLI

The Vault command used to create a managed key from a manually generated key on the HSM is almost identical to the command that we used to dynamically generate a key on the HSM in the Testing PKI Operations section.

1. Manually generate an 2048 bit RSA key in Vault with the key label "VaultManualKey".

```
$ vault write /sys/managed-keys/pkcs11/hsm-key library=hsm1 token_label=Futurex pin=$PIN key_
label="VaultManualKey" allow_generate_key=false allow_store_key=false mechanism=0x0001 key_
bits=2048 any_mount=false
```

Note: In the **key_label** field, we're specifying the PKCS11 label that we assigned to the key using the **keytable extdata** FXCLI command in the previous section. The main difference you should note in the above command is that we set **allow_generate_key** to **false** to tell Vault not to attempt to generate a key on the HSM if it cannot find the key we're referencing.

2. Verify that the key configuration has been written to Vault.

```
$ vault list /sys/managed-keys/pkcs11
```

3. Verify that the key configuration is valid by test signing some data.

```
$ vault write -f /sys/managed-keys/pkcs11/hsm-key/test/sign
```

APPENDIX B: 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