DOGTAG CERTIFICATE SYSTEM

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 Dogtag Certificate System using Futurex PKCS #11 libraries. For additional questions related to your HSM, see the relevant user guide.

[1.2] APPLICATION DESCRIPTION

[1.2.1] About Dogtag Certificate System

The Dogtag Certificate System is an enterprise-class open source Certificate Authority (CA). It is a full-featured system, and has been hardened by real-world deployments. It supports all aspects of certificate lifecycle management, including key archival, OCSP and smartcard management, and much more. The Dogtag Certificate System can be downloaded for free, and extensive documentation is available at the Dogtag PKI GitHub Wiki.

[1.2.2] Key Features

Dogtag is a collection of technologies that allow enterprises to deploy PKI on a large scale. It has features such as:

- Certificate issuance, revocation, and retrieval
- Certificate Revocation List (CRL) generation and publishing
- Certificate Profiles
- Simple Certificate Enrollment Protocol (SCEP)
- Local Registration Authority (LRA) for organizational authentication and policies
- Encryption key archival and recovery
- Smartcard lifecycle management
    - Token profiles
    - Token enrollment, on-hold, key recovery, and format
    - Face-to-face enrollment with the security officer workstation interface

For more, see the Features page on the Dogtag PKI GitHub Wiki.

[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] FUTUREX CERTIFICATION PROCESS

The Futurex Certification Process is a rigorous and standardized approach to testing and certifying integrations between third-party applications and Futurex's HSMs and key management servers (i.e., KMES Series 3). The certification process is designed to ensure that third-party application integrations are fully tested and validated in a lab environment before they are deployed in a production environment. Futurex's Integration Engineering team implements this process so that customers can have confidence that third-party applications will integrate seamlessly with Futurex's HSMs and KMES Series 3 devices, and that all operations will result in the expected behavior. The certification process involves several steps, including research, testing, troubleshooting, and certification, and is fully documented in an integration guide for each integration. The full process is outlined below:

1. Research the third-party application to gain a general understanding of the solution and the protocol it uses to integrate with an HSM or KMS device (i.e., PKCS #11, Microsoft CNG, JCE, OpenSSL Engine, KMIP).
2. Determine the scope of the third-party application's use of the HSM or KMS device, including the specific functionalities it utilizes (i.e., data encryption, key protection, entropy, etc.).
3. Install and configure the third-party application in a lab environment, where all testing and validation will take place.
4. Establish a connection between the third-party application and the Futurex device, which typically involves configuring TLS certificates and creating roles and identities that the third-party application will use to connect and authenticate to the Futurex device.
5. Initiate a request from the third-party application to the Futurex device, such as generating keys or certificates, encrypting or decrypting data, or other cryptographic functions.
6. If any errors occur during the testing process, the Integration Engineering team will diagnose the issues and take necessary corrective actions. If necessary, the team will also document the error(s) by creating engineering change requests (ECRs) to ensure all issues are addressed and resolved before certification.
7. After any necessary engineering changes have been made, a new end-to-end test will be performed to ensure that all errors have been resolved and that all operations are successful.
8. Certify the integration by creating an integration guide that covers all necessary prerequisites, configurations required in both the third-party application and the Futurex device, and how to test the functionality.

Overall, following these steps helps ensure that the integration between the third-party application and the Futurex device is fully tested and validated, and that any errors or issues are resolved before the integration is certified as fully supported.

[3] PREREQUISITES

Supported Hardware:

- Vectera Plus, 6.7.x.x and above

Recommended Operating Systems:

- Fedora 28

Note: Dogtag Certificate System is primarily meant to be installed on the Fedora Linux operating system. Futurex offers versions of the Futurex PKCS #11 library for both RHEL 7 and RHEL 8. It is recommended to install Dogtag on a Fedora 28 system because RHEL 8 is centered around Fedora 28. If you plan to install Dogtag on a non-Fedora system, you will need to manually build the system by following the instructions at the following link: https://github.com/dogtagpki/pki

[4] INSTALL FUTUREX PKCS #11 (FXPKCS11)

In a Linux environment, you must download a tarball of the **Futurex PKCS #11 (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 section provides step-by-step installation instructions.

Note: You need to install FXPKCS11 on the same computer where Dogtag Certificate System is installed.

[4.1] 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, you can install it in $HOME/bin.

The extracted content of the tar file is a single **fxpkcs11** directory. Inside the fxpkcs11 directory are 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 stored 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.

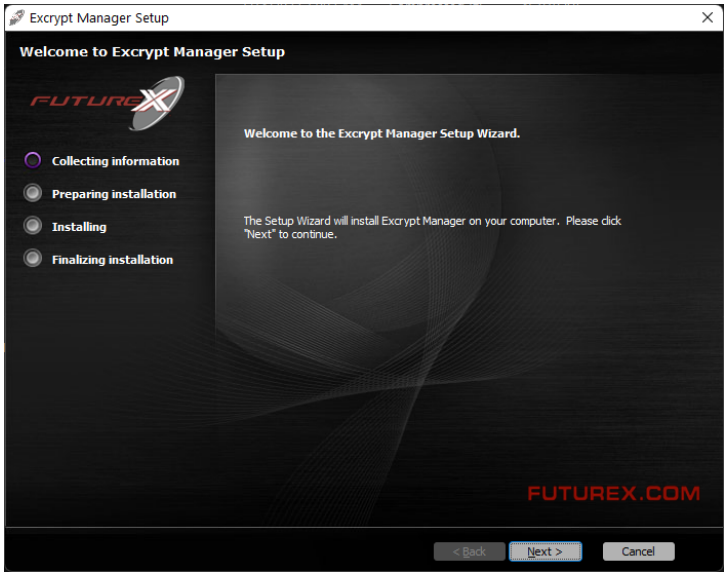


FIGURE: EXCRYPT MANAGER SETUP WIZARD

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.

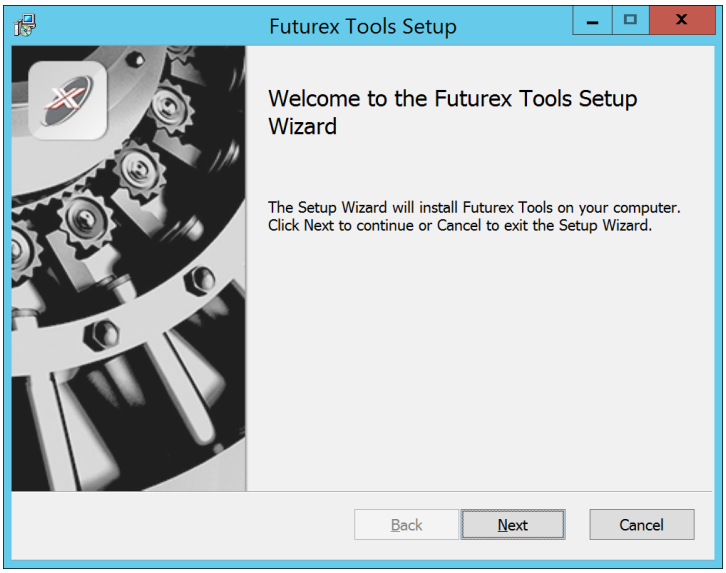


FIGURE: FUTUREX TOOLS SETUP WIZARD

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

You must perform several configuration items in order to establish a connection between the Futurex PKCS #11 library and the Vectera Plus. These items are the following:

Note: You can complete all of the steps in this section through either Excrypt Manager or FXCLI (if using a physical HSM rather than a virtual HSM). Optionally, yo can complete steps 4 through 6 through the Guardian Series 3 (Please refer to the applicable guide for configuring HSMs with the Guardian Series 3).

1. Connect to the HSM via the front USB port. If you are using a virtual HSM for the integration you will have to connect to it over the network either via FXCLI, the Excrypt Touch, or the Guardian Series 3
    a. Connecting via Excrypt Manager
    b. Connecting via FXCLI
2. Validate the correct features are enabled on the HSM
3. Setup 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 Application Partition created in the previous step
7. Configure TLS Authentication. There are two options for this:
    a. Enabling server-side authentication
    b. Creating client certificates for mutual authentication

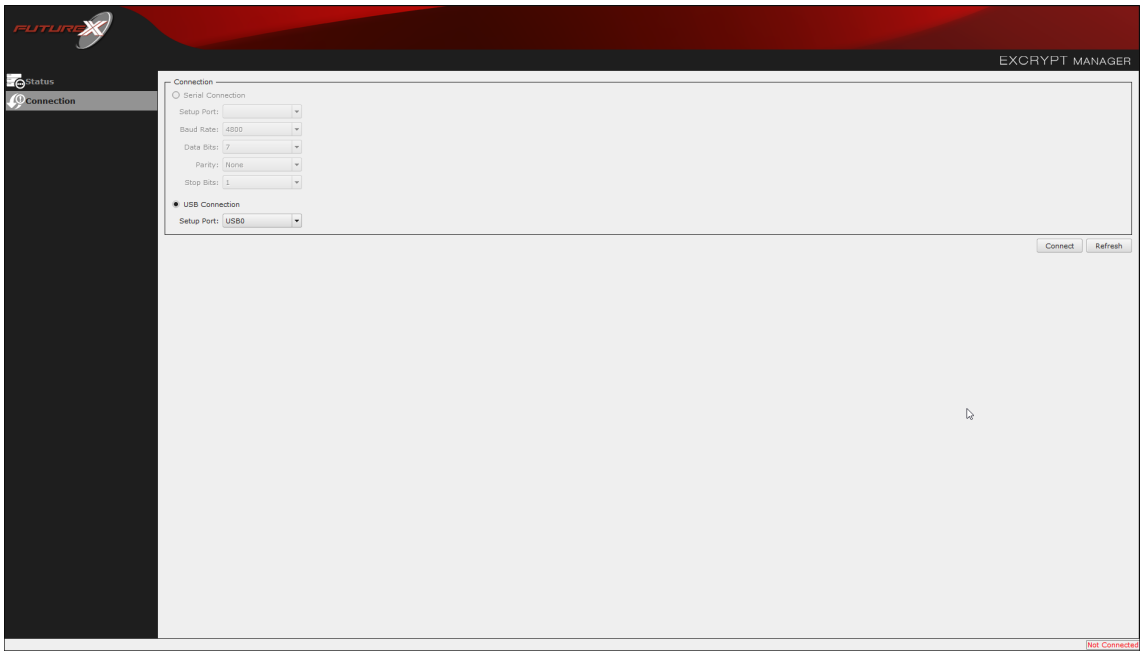
Each of these action items is detailed in the following subsections.

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

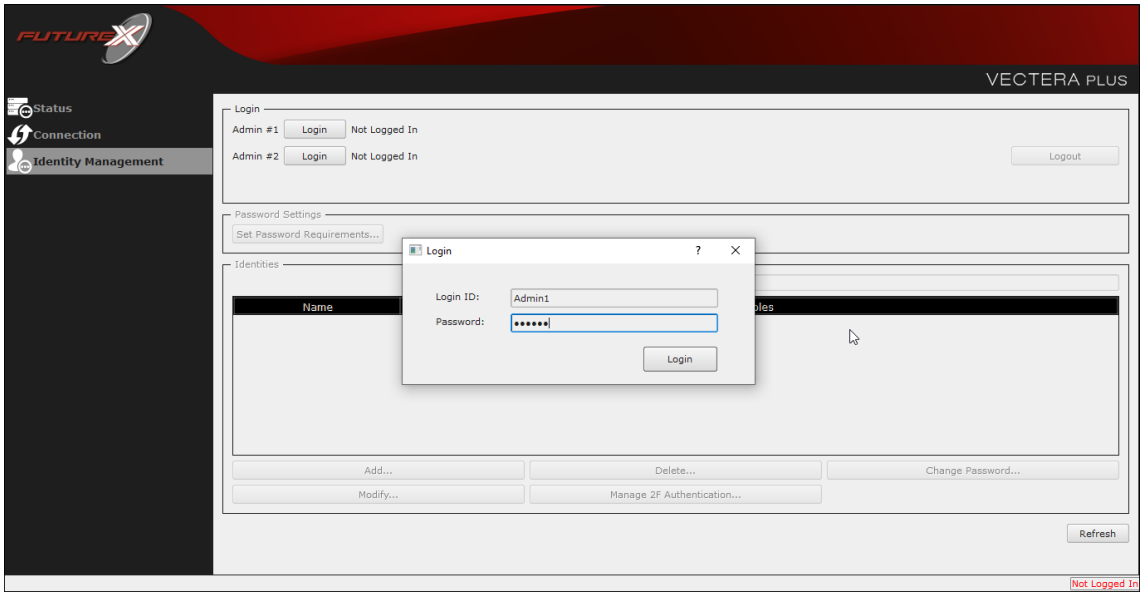
For both Excrypt Manager and FXCLI you need to connect your laptop to the front USB port on the HSM.

Connecting via Excrypt Manager

Open Excrypt Manager, click **Refresh** in the lower right-hand side of the Connection menu. Then select **USB Connection**>**Connect**.

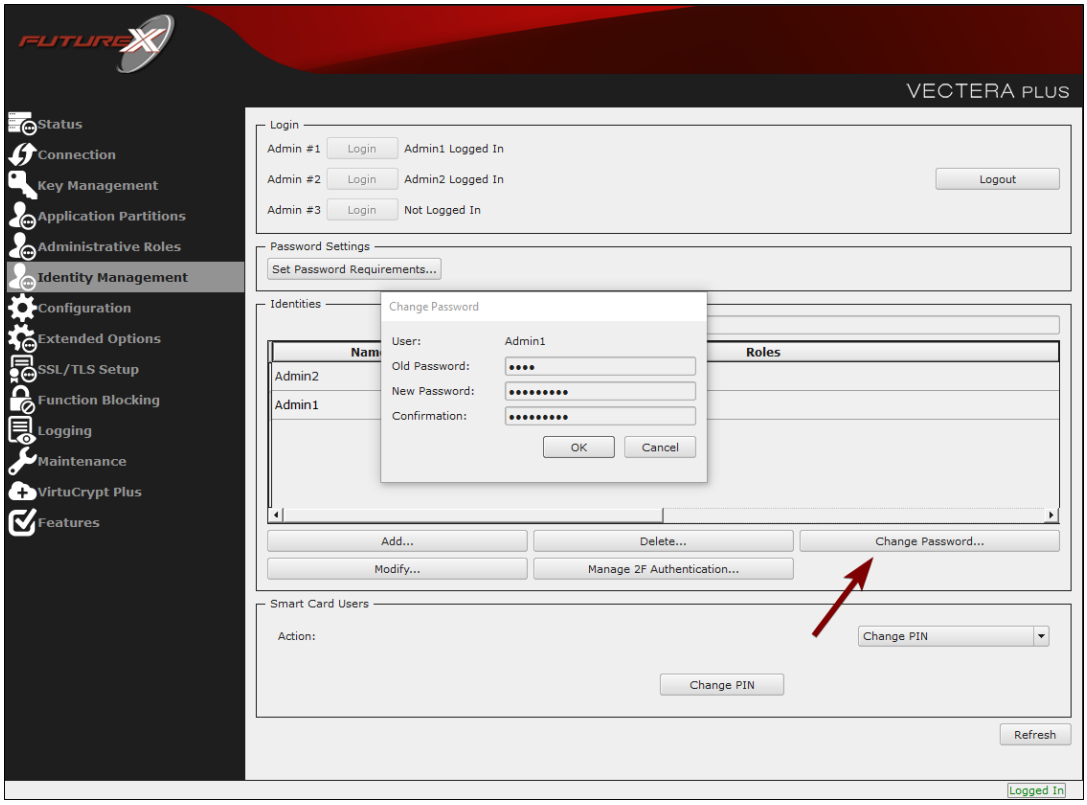


Log in with both default Admin identities.



You must change the default Admin passwords (i.e. **safe**) for both of your default Admin Identities (e.g. **Admin1** and **Admin2**) in order 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 (e.g. **Admin1**), and click **Change Password…**. Enter the old password and enter the new password twice. Click **OK**. Perform the same steps as above for the second default Admin identity (e.g. **Admin2**).



Connecting via FXCLI

Open the FXCLI application and run the following commands:

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

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

You must change the default Admin passwords (i.e. “safe”) for both of your default Admin Identities (e.g. **Admin1** and **Admin2**) in order to load the major keys onto the HSM.

The following FXCLI commands can be used to change the passwords for each default Admin Identity.

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

Note: The user change-password commands above will prompt you to enter the old and new passwords. It is necessary to run the command twice (as shown above) because the default password must be changed for both default Admin identities.

[7.2] FEATURES REQUIRED IN HSM

In order to establish a connection between the PKCS #11 Library and the Futurex HSM, the HSM must be configured with the following features:

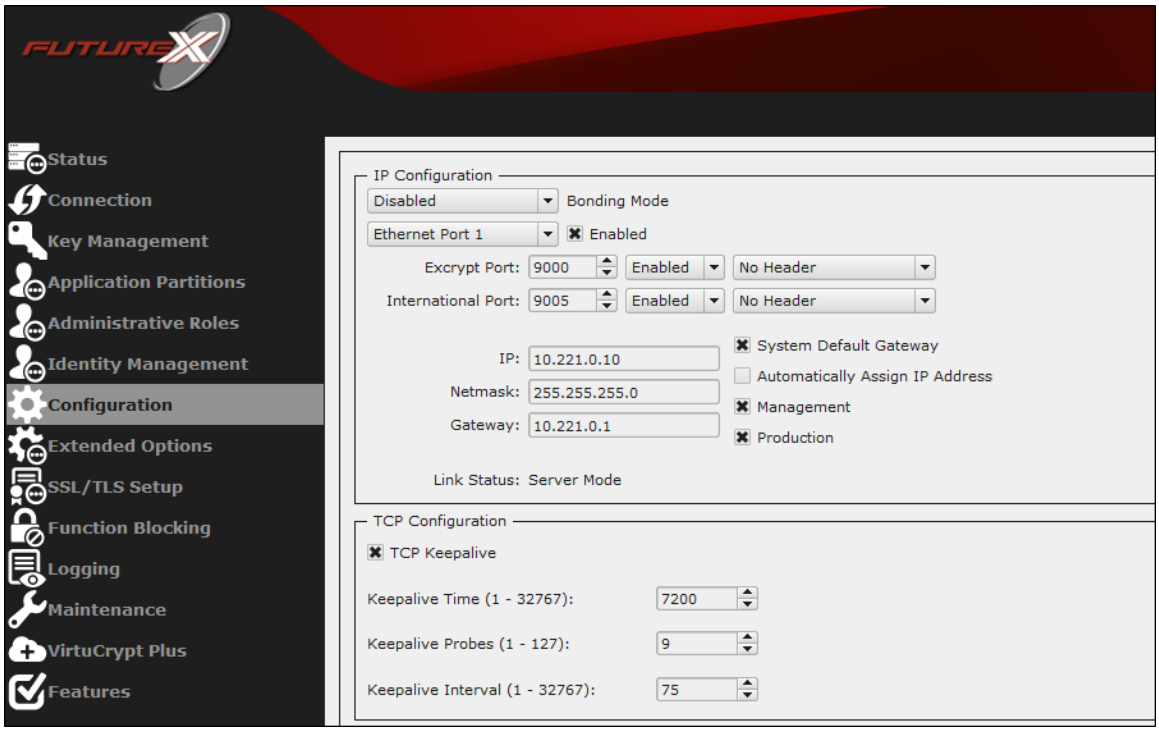
- **PKCS #11** - Enabled
- **Command Primary Mode** - General Purpose (GP)

Note: For additional information about how to update features on your HSM, please refer to your HSM Administrator's Guide, section **"Download Feature Request File"**. You can enable the option to create the FTK major key in the HSM with the command .You are required to use this key in order to use the PKCS #11 library to communicate with the HSM. For detailed information about how to load major keys in HSMs please refer to your HSM Administrator's Guide.

[7.3] NETWORK CONFIGURATION (HOW TO SET THE IP OF THE HSM)

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

Open **Configuration**. You will see the option to modify the IP configuration, as shown below:



Alternatively, you can use the following FXCLI command to set the IP for the HSM:

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

Note: The following should be considered at this point:

- You can complete all of the remaining HSM configurations in this section using the Guardian Series 3 (please refer to Appendix A for instructions on how to do so), with the exception of the final subsection that covers how to create connection certificates for mutual authentication.
- If you are performing the configuration on the HSM directly now but plan to add the HSM to a Guardian later, you may have to synchronize the HSM after it is added to a Device Group on the Guardian.
- If you use-case requires configuration through a CLI then you should manage the HSMs directly.

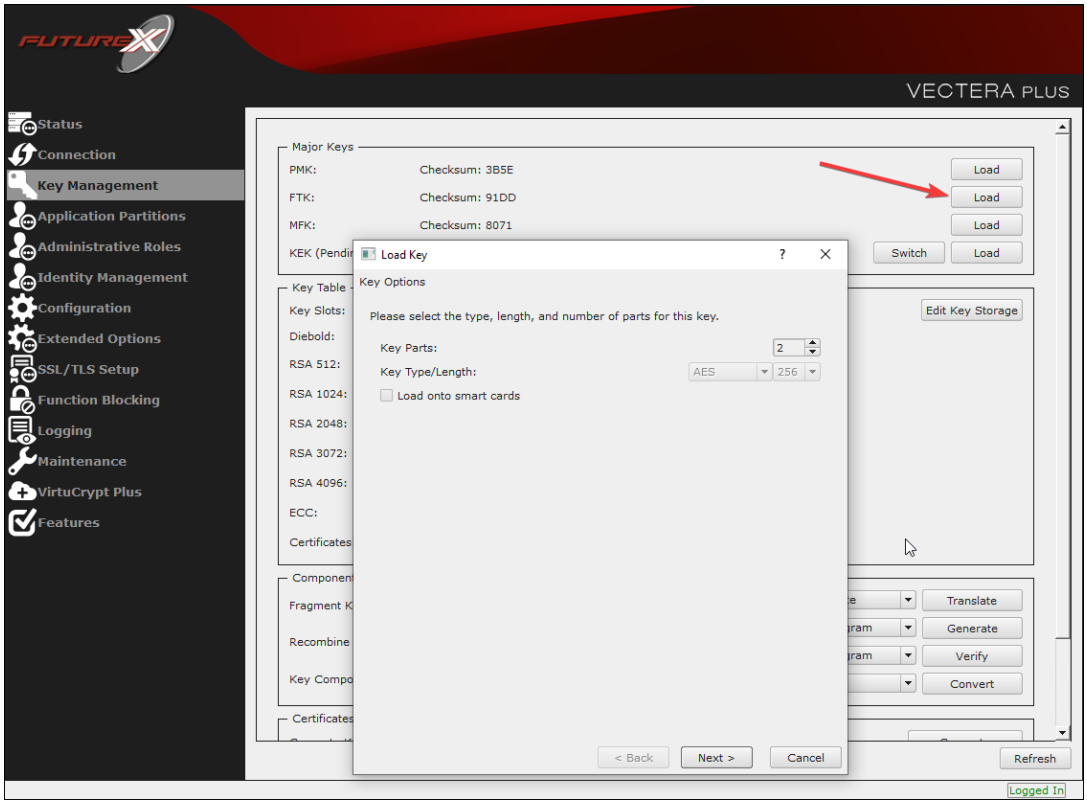
[7.4] LOAD FUTUREX KEY (FTK)

For this step you must log 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 he same FTK for syncing HSMs. An HSM must have an FTK before you can use it with PKCS #11, it must have an FTK.

Note: You can also complete this process using FXCLI, the Excrypt Touch, or the Guardian Series 3. For more information about how to load the FTK into an HSM using these tools/devices, please see the relevant Administrative Guide.

After logging in, select **Key Management** > **Load** under FTK. 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.



Alternatively, you can use the following FXCLI commands to load an FTK onto 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.

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

If it's a second HSM that you're setting up in a cluster then 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

For this step you must log 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 you can consider the terms "Application Partition" and "Role" to be synonymous.

Configure a Transaction Processing Connection

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

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

Go to **Application Partitions**, select the **Transaction Processing Application Partition**, and click **Modify**.

Under the **Permissions** tab, leave the top-level **Keys** permission checked, but uncheck the **All Slots** sub permission.

Under the **Key Slots** tab you need to ensure that there are no key ranges specified. By default, the Transaction Processing Application Partition has access to the entire range of key slots on the HSM.

Under 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)

Alternatively, the following FXCLI commands can be used to remove all permissions and key ranges that are currently assigned to the **Transaction Processing** role and enable only the PKCS #11 Communication commands:

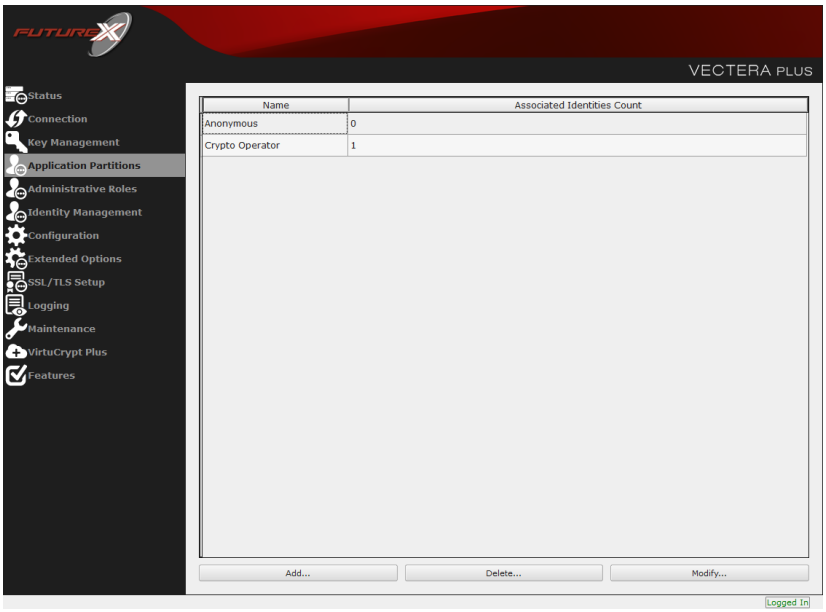
```
$ role modify --name Anonymous --clear-perms --clear-key-ranges
```

```
$ 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
```

Create an Application Partition

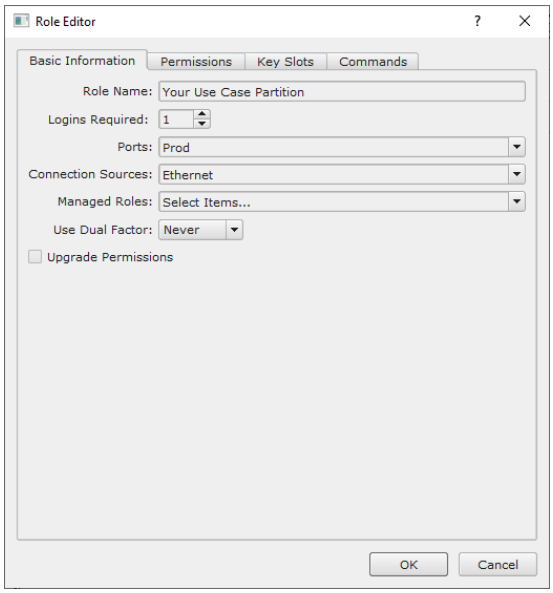
In order to segregate applications on the HSM, you must create an Application Partition specifically for your use case. Application partitions are used to segment the permissions and keys on an HSM between applications. The process for configuring a new application partition is outlined in the following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

Open **Application Partitions** > **Add**.



Fill in all of the fields in **Basic Information** exactly how you see below (except for the **Role Name** field). In the **Role Name** field, specify any name that you would like for this new Application Partition. Set **Logins Required** to *1*. Set **Ports** to *Prod*. Configure **Connection Sources** to *Ethernet*. Leave **Managed Roles** blank because we'll be specifying the exact Permissions, Key Slots, and Commands that we want this Application Partition/Role to have

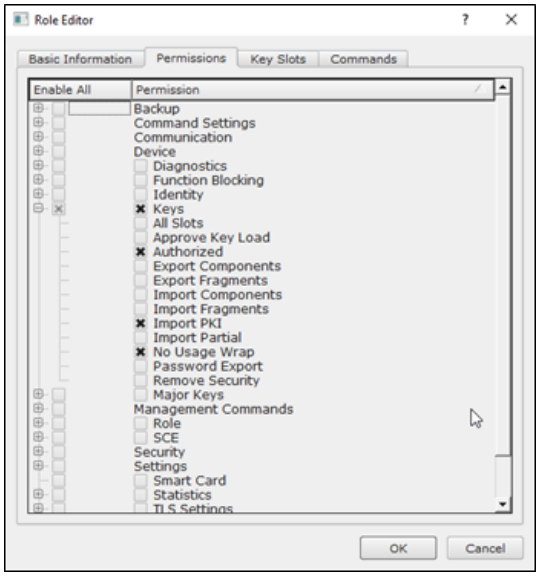
access to. Set **Use Dual Factor** to *Never*.



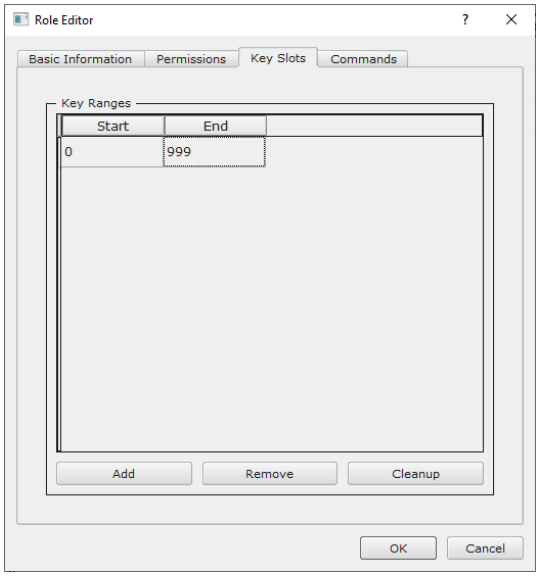
Under **Permissions**, select the following key permissions:

- **Keys**
- **Authorized**
- **Import PKIs**
- **No Usage Wrap**

The **Authorized** permission allows for keys that require login. The **Import PKI** permission allows trusting an external PKI, which is used by some applications to allow for PKI symmetric key wrapping (It is not recommended to enable unless using this use case). The **No Usage Wrap** permission allows for interoperable key wrapping without defining key usage as part of the wrapped key (We only recommend this if exchanging keys with external entities or using the HSM to wrap externally used keys).



Under key slots, we recommend you create a range of 1000 total keys (here we've specified the key range 0-999) that do not overlap with another Application Partition. Within this range, there must be ranges for both symmetric and asymmetric keys. If the application requires more keys, configure accordingly.



There are particular functions based on application requirements that you must enable on the Application Partition in order to use the HSM's functionality. The commands that need to be enabled for the Dogtag integration are listed below. These can be enabled under **Commands**.

PKCS #11 Communication Commands

- **ECHO**: Communication Test/Retrieve Version
- **HASH**: Retrieve device serial
- **GPKM**: Retrieve key table information
- **GPKR**: General purpose key settings get (read-only)
- **GPKS**: General purpose key settings get/change
- **RAND**: Generate random data
- **TIME**: Set Time

Key Operations Commands

- **GPCA**: General purpose add certificate to key table
- **GPKD**: General purpose key slot delete/clear
- **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

Alternatively, you can use the following FXCLI commands to create the new Application Partition and enable all needed functions:

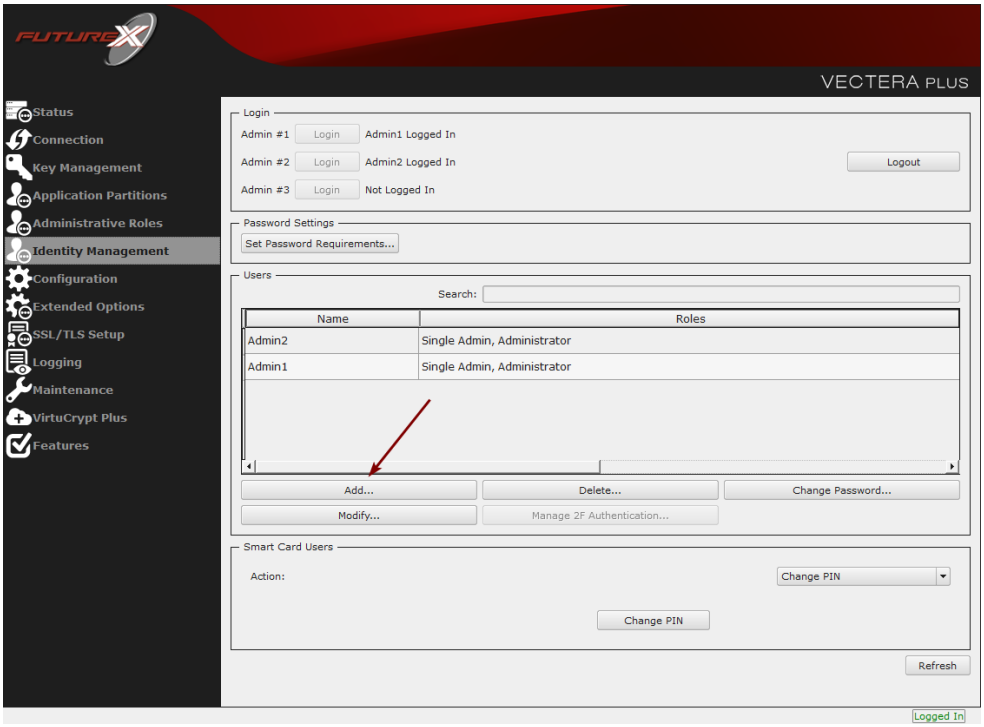
```
$ role add --name Role_Name --application --key-range (0,999) --perm "Keys:Authorized" --perm "Key-
s:Import PKI" --perm "Keys:No Usage Wrap"
```

```
$ role modify --name [role_name] --clear-perms --add-perm Excrypt:ECHO --add-perm Excrypt:HASH --
add-perm Excrypt:GPKM --add-perm Excrypt:GPKR --add-perm Excrypt:GPKS --add-perm Excrypt:GPCA --
add-perm Excrypt:GPKD --add-perm Excrypt:GRSA --add-perm Excrypt:LRSA --add-perm Excrypt:GPSR --
add-perm Excrypt:RAND --add-perm Excrypt:TIME
```

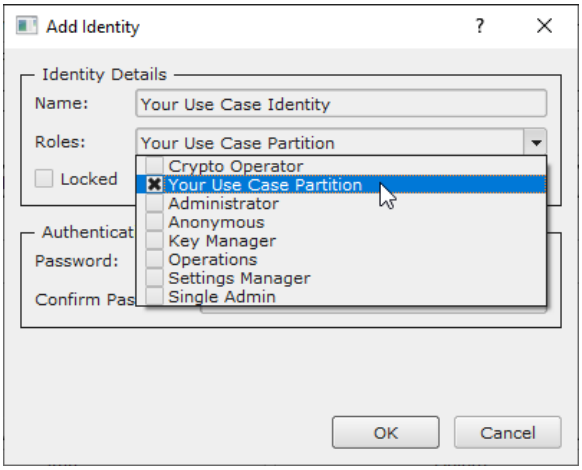
[7.6] CREATE NEW IDENTITY AND ASSOCIATE IT WITH THE NEWLY CREATED APPLICATION PARTITION

For this step you must log in with an identity that has a role with permissions **Identity:Add**. You can use the default Administrator role and Admin identities.

Create a new identity associated with the Application Partition created in the previous step. To create this new identity, click **Identity Management** > **Add**.



Specify a name for the new identity and open the **Roles** dropdown to select the name of the Application Partition created in the previous step. This will associate the new Identity with the Application Partition that you created.



Alternatively, you can use the following FXCLI to create a new Identity and associate it with the role that you created:

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

This new identity must be set 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>
```

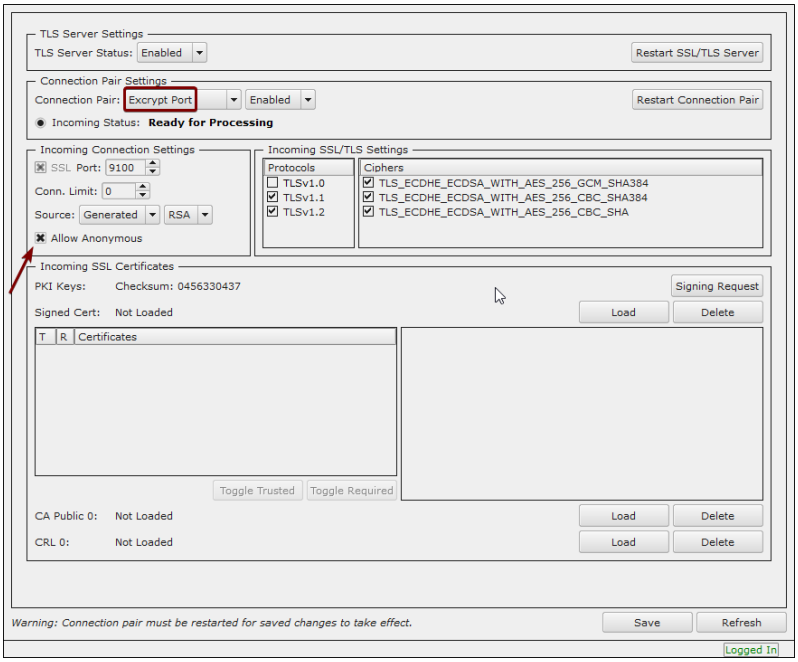
Note: Crypto Operator in the fxpkcs11.cfg file must match the name of the identity created in the HSM.

[7.7] CONFIGURE TLS AUTHENTICATION

For this step you must log 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.

Enable Server-Side Authentication (Option 1)

We recommend mutually authenticating to the HSM using client certificates but server-side authentication is also supported. To enable server-side authentication click **SSL/TLS Setup** > **Excrypt Port** > **Allow Anonymous**.



Alternatively, you can use the following FXCLI command to enable server-side authentication with the **Allow Anonymous** SSL/TLS setting:

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

Create Connection Certificates for Mutual Authentication (Option 2)

We recommend mutually authenticating to the HSM using client certificates, which the system enforces by default. In the example below, the FXCLI generates a CA that signs the HSM server certificate and a client certificate. The Windows PowerShell with OpenSSL generates the client keys and CSR. For other options for managing certificates required for mutual authentication with the HSM, please review the relevant Administrator's guide.

Find the FXCLI program you installed with FXTools, and run it as an administrator.

Things to note:

- For this example, you must connect the computer running FXCLI to the front port of the HSM. However, you can use the HSMs Web Portal, or the Excrypt Touch for remote management.
- If you do not specify a file path for commands that create an output file, FXCL will save the file to the current directory.
- Using user-generated certificates requires a PMK to be loaded on the HSM.
- If you run **help** by itself it will show a full list of available commands. You can see all of the available options for any given command by running the command name followed by **help**.

```
# 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 pass-
word. You will need to run this command twice.
$ login user
```

```
# Generate 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 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
```

Note: You must run the following OpenSSL commands from Windows PowerShell rather than from the FXCLI program.

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

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

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
```

Switch back to Windows PowerShell for the remaining commands.

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

[8] 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**.

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

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**.

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.

[8.1] SPECIAL DEFINES REQUIRED FOR THIS INTEGRATION

For this integration, the following two defines must be added to the **<CONFIG>** section in the FXPKCS11 configuration file:

```
<FORCED-ASYMMETRIC-USAGE>    SIGN | VERIFY        </FORCED-ASYMMETRIC-USAGE>
<CHECK-ALREADY-LOGGED-IN>    NO                   </CHECK-ALREADY-LOGGED-IN>
```

[9] DOGTAG CERTIFICATE SYSTEM INSTALLATION AND SUBSYSTEM DEPLOYMENT

This section will guide you through the installation process of Dogtag Certificate System on Fedora Linux 28. Our example will demonstrate the installation of a CA subsystem using a self-signed CA signing certificate, with the certificates and their corresponding keys being securely stored on the Futurex Vectera Plus HSM.

[9.1] INSTALLING THE DOGTAG PACKAGES

1. Install the Dogtag packages using the following command:

```
sudo dnf install pki-ca pki-kra 389-ds-base
```

[9.2] CREATING THE DIRECTORY SERVER INSTANCE FOR THE DOGTAG INTERNAL DB

The Dogtag CA and KRA subsystems use a 389 Directory Server as an internal database. To configure one, follow the steps below:

1. Run the following command to login as the root user:

```
sudo su
```

2. Set a FQDN (Fully qualified domain name) as the hostname for your Fedora 28 system.
   a. Edit the **/etc/hosts** file as follows:

      Note: You can set any valid FQDN. It does not need to be set to pki.example.com.

```
vim /etc/hosts
```

```
127.0.0.1   pki.example.com
::1         pki.example.com
```

   b. You must also run the command below to update the hostname in the **/etc/hostname** file.

```
hostnamectl set-hostname pki.example.com
```

3. Create a directory for storing the 389 Directory Server configuration file:

```
mkdir -p /etc/389-ds
```

4. Create a configuration file in the new directory with the **[General]** and **[slapd]** sections configured as follows:

```
vim /etc/389-ds/setup.inf
```

```
[General]
FullMachineName= pki.example.com
SuiteSpotUserID= nobody
SuiteSpotGroup= nobody

[slapd]
ServerPort= 389
ServerIdentifier= pki-tomcat
Suffix= dc=example,dc=com
RootDN= cn=Directory Manager
RootDNPwd= Password
```

5. Run the directory server installation script, selecting the defaults or customizing as desired:

```
setup-ds.pl --silent --file=/etc/389-ds/setup.inf
```

[9.3] RUN THE PKISPAWN SCRIPT TO CREATE AND CONFIGURE A SUBSYSTEM INSTANCE

The **pkispawn** command line tool is used to install and configure a new PKI instance. It eliminates the need for separate installation and configuration steps, and may be run either interactively, as a batch process, or a combination of both (batch process with prompts for passwords). Refer to the pkispawn man page for detailed information about all supported options by running "man pkispawn".

The pkispawn command reads in its default installation and configuration values from a plain text configuration file (/etc/pki/default.cfg). This file consists of name=value pairs divided into [DEFAULT], [Tomcat], [CA], [KRA], [OCSP], [TKS], and [TPS] sections.

Note: It is strongly recommended that you read the full documentation to understand the purpose of every parameter in the /etc/pki/default.cfg file. This will allow you to customize your PKI environment to your specific needs.

Dogtag's recommended procedure for spawning a subsystem that uses an HSM is to create an override configuration file that contains only the parameters necessary for using the HSM as its token. Any parameter settings in this file will override the parameter settings in the default.cfg file.

Any of the various Dogtag PKI subsystems (CA, KRA, OCSP, TKS, TPS) can be spawned to use the HSM, but this integration guide will focus solely on the Certificate Authority (CA) for brevity.

Prepare an override configuration file with required HSM parameters

1. Create the override configuration file for the CA subsystem.

```
sudo vim ca.cfg
```

The following is an example override file that can be used for spawning a CA subsystem with the HSM:

Note: All values contained within angle brackets need to be set to a specific value by the user. All other values should be set exactly as shown.

Note: The pki_ds_password value must match the password set for the directory manager when 389 Directory Server was installed.

```
[DEFAULT]
############################
# Provide HSM parameters #
############################
pki_hsm_enable=True
pki_hsm_libfile=<path_to_fxpkcs11_libfile>
pki_hsm_modulename=FxPKCS11
pki_token_name=Futurex
pki_token_password=<hsm_identity_password>

######################################
# Provide PKI-specific HSM token names #
######################################
pki_audit_signing_token=Futurex
pki_ssl_server_token=Futurex
pki_subsystem_token=Futurex

################################
# Provide PKI-specific passwords #
################################
pki_admin_password=<pki_admin_password>
pki_client_pkcs12_password=<pki_client_pkcs12_password>
pki_ds_password=<pki_ds_password>

####################################
# Provide non-CA-specific passwords #
####################################
pki_client_database_password=<pki_client_database_password>

[CA]
######################################
# Provide CA-specific HSM token names #
######################################
pki_ca_signing_token=Futurex
pki_ocsp_signing_token=Futurex
```

After you have finished editing, save the file.

Run the pkispawn utility

1. In a terminal, run the following command to deploy a CA subsystem using the Vectera Plus HSM:

   Note: The full path to the ca.cfg file is required if you are not running the command from the directory where the ca.cfg file is saved.

   ```
   sudo pkispawn -f ca.cfg -s CA -vvv
   ```

   Note: You will most likely see a warning message about manually adding a module while p11-kit is enabled. You can disregard this warning and press <enter> to continue.

If the deployment is successful, an installation summary similar to the following will be presented after the command completes:

```
==========================================================================
                              INSTALLATION SUMMARY
==========================================================================

    Administrator's username:             caadmin
    Administrator's PKCS #12 file:
        /root/.dogtag/pki-tomcat/ca_admin_cert.p12

    To check the status of the subsystem:
        systemctl status pki-tomcatd@pki-tomcat.service

    To restart the subsystem:
        systemctl restart pki-tomcatd@pki-tomcat.service

    The URL for the subsystem is:
        https://pki.example.com:8443/ca

    PKI instances will be enabled upon system boot

==========================================================================
```

Note: If the **pkispawn** command fails, you need to run the following command to delete the subsystem instance that was only partially created before re-attempting to run pkispawn.

```
sudo pkidestroy -s CA -i pki-tomcat
```

[9.4] VIEW THE KEYS AND CERTIFICATES THAT DOGTAG CREATED ON THE HSM

To view the keys and certificates that Dogtag created on the HSM, we will use the **PKCS11Manager** utility packaged with the Futurex PKCS #11 module.

1. In a terminal, navigate to the directory where the FXPKCS11 module is installed (e.g., /usr/local/bin/fxpkcs11) and run PKCS11Manager using the following command:

   ```
   ./PKCS11Manager
   ```

   This will present the following main menu:

   ```
   Main Menu
       1. Print Library/Token Info

       2. Login
       3. Logout

       4. Generate Key

       5. Find Objects
       6. Modify Objects
       7. Delete Objects

       8. Generate Random Data

       9. Sign Data
       10. Verify Data
   ```

```
11. Wrap Key
12. Unwrap Key

13. Import public key

0. Exit
```

2. Type "2" to login, then press Enter.
3. Type "1", then press Enter.
4. Type the password of the identity that is defined in the FXPKCS11 configuration file, then press Enter.

   If successful, you will receive confirmation that you are logged in.
5. Type "5" to find objects, then press Enter.
6. Type "1" to find all objects, then press Enter.

   Information will be printed for all keys and certificates that the connecting identity has access to.

   Note: Dogtag PKI creates 15 objects on the HSM for a CA subsystem deployment.

[9.5] IMPORT THE CA ADMINISTRATOR PKCS #12 FILE INTO THE BROWSER

Note: The following steps were completed using a Firefox web browser. There may be some differences in the steps when using a different browser, but the overall intent of the process is the same.

1. In Firefox, navigate to **Preferences** > **Privacy & Security** > **Certificates** and click the **View Certificates** button.
2. Under the **Your Certificates** tab, select **Import** to import the CA Administrator PKCS #12 file (i.e., ca_admin_cert.p12). When it prompts for a password, enter the value that was configured for the **pki_client_pkcs12_password** define in the ca.cfg file in section 9.3.

   Note: The location of the ca_admin_cert.p12 file was included in the installation summary for the CA subsystem deployment.

[9.6] ACCESSING THE NEW CA SUBSYSTEM IN THE BROWSER

1. Access the Dogtag Certificate System subsystem console by navigating to the URL below:

   https://<fully qualified domain name>:8443/pki/ui/

Note: When submitting Certificate Signing Requests (CSRs) in Dogtag Certificate System, the **Common Name** and **UID** fields are both required. If you submit a request with only the **Common Name** field completed, the request will fail, and you will receive an error stating that the **Subject Name** does not match.

This completes the Dogtag Certificate System integration with the Futurex Vectera Plus HSM. All CA subsystem keys are secured within an application partition on the Vectera Plus and available to Dogtag Certificate System when required.

APPENDIX A: XCEPTIONAL SUPPORT

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

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

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

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

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

ENGINEERING CAMPUS

864 Old Boerne Road

Bulverde, Texas, USA 78163

Phone: +1 830-980-9782

+1 830-438-8782

E-mail: info@futurex.com

XCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

SOLUTIONS ARCHITECT

E-mail: solutions@futurex.com