JAVA KEYTOOL

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 Java Keytool integration with Futurex HSMs. For additional questions related to your HSM, see the relevant user guide or reach out to a Solutions Architect for help.

[1.1.1] About Keytool Integration

Keytool is a Java program that allows Java application developers to obtain and manage CA certificates for authentication and digital signing, along with the KeyStore of keys and certificate chains. Java Keytool integration with Futurex HSMs is made simple by the PKCS #11 library. For added hardware-backed security, Keytool supports integration with HSMs for managing the key lifecycle in a FIPS-140 Level 3 cryptographic boundary.

[1.2] COPYRIGHT AND TRADEMARK NOTICES

[1.3] TERMS OF USE

[1.4] GUARDIAN INTEGRATION

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

- Centralize device management
- Eliminates points of failure
- Distribute transaction loads

- Group-specific function blocking
- User-defined grouping systems

Please see applicable guide for configuring HSMs with the Guardian Series 3.

[2] PREREQUISITES

Supported Hardware:

- Vectera Plus, 6.7.x.x and above

Supported Operating Systems:

- Windows 7 and above
- Linux (Ubuntu, Debian and Red Hat-based distributions)

Other:

- Java 7, 8, or 9
- OpenSSL

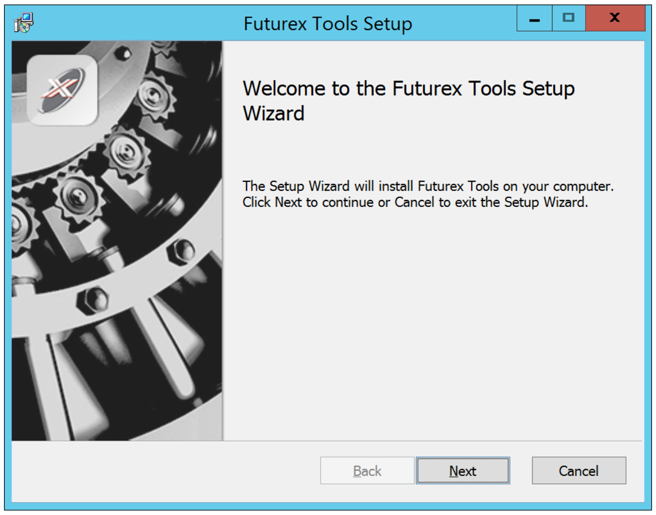
[3] INSTALL FUTUREX PKCS #11 (FXPKCS11)

In a Windows environment, the easiest way to install the Futurex PKCS #11 (FXPKCS11) module is through installing **FXTools**. FXTools can be downloaded from the Futurex Portal. In a Linux environment, you need to download a tarball of the PKCS #11 binaries from the Futurex Portal. Then, extract the .tar file locally where you want the application to be installed in your file system. Step by step installation instructions for both of these scenarios is provided in the following subsections.

NOTE: The Futurex PKCS #11 module needs to be installed on the server that will be using the HSM.

[3.1] INSTRUCTIONS FOR INSTALLING THE FXPKCS11 MODULE USING FXTOOLS IN WINDOWS

- Run the FXTools installer as an administrator


Futurex Tools Setup

Welcome to the Futurex Tools Setup Wizard

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

FIGURE: FUTUREX TOOLS SETUP WIZARD

By default, all tools are installed on the system. A user can overwrite and choose not to install certain modules.

- **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 –**The client used to connect a Futurex Excrypt Touch to a local laptop, via USB, and a remote Futurex device.

After starting the installation, all noted services are installed. If the Futurex Secure Access Client was selected, the Futurex Excrypt Touch driver will also be installed (Note this sometimes will start minimized or in the background).

After installation is complete, 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, located in their corresponding directory with a *.cfg* extension. In addition, the CNG and CSP Modules are registered in the Windows Registry (*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider*) and are installed in the "*C:\Windows\System32\*" directory.

[3.2] INSTRUCTIONS FOR INSTALLING THE PKCS #11 MODULE IN LINUX

Extract the appropriate tarball file for your specific Linux distribution in the desired working directory.

NOTE: For the Futurex PKCS #11 module to be accessible system-wide, it would need to be placed into */usr/local/bin* by an administrative user. If the module only needs to be utilized by the current user, then installing into *$HOME/bin* would be the appropriate location.

The extracted content of the *.tar* file is a single *fxpkcs11* directory. Inside of the *fxpkcs11* directory are the following files and directories (Only files/folders that are relevant to the installation process are included below):

- *fxpkcs11.cfg* -> PKCS #11 configuration file
- *x86/* - This folder contains the module files for 32-bit architecture
- *x64/* - This folder contains the module files for 64-bit architecture

Within the *x86* and *x64* directories are two directories. One named *OpenSSL-1.0.x* and the other named *OpenSSL-1.1.x*. Both of these OpenSSL directories contain the PKCS #11 module files, built with the respective OpenSSL versions. These files are listed below, with short descriptions of each:

- *configTest* -> Program to test configuration and connection to the HSM
- *libfxpkcs11.so* -> PKCS #11 Library File
- *PKCS11Manager* -> Program to test connection and manage the HSM through the PKCS #11 library

The *configTest* and *PKCS11Manager* programs look for the *fxpkcs11.cfg* file at the following path:

```
/etc/fxpkcs11.cfg
```

Because of this, it is necessary either to move the *fxpkcs11.cfg* file from the */usr/local/bin/fxpkcs11* directory to the */etc* directory, or to set the FXPKCS11_CFG environment variable to point to the *fxpkcs11.cfg* file.

[4] INSTALL EXCRYPT MANAGER (IF USING WINDOWS)

The following two sections will cover how to install the **Excrypt Manager** and **FXCLI** applications. These tools are used to configure the HSM in subsequent sections. Note that installing Excrypt Manager is optional, but installing FXCLI is required, as FXCLI is the method that is used for configuring TLS mutual authentication between the Vectera Plus and the application that is being integrated.

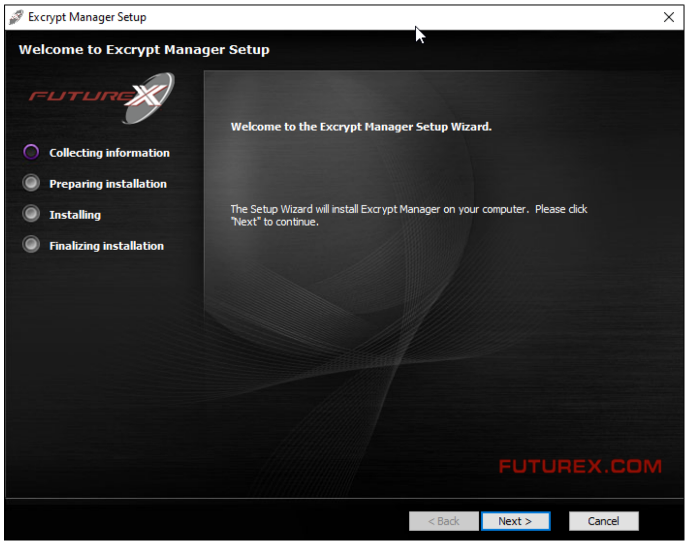
NOTE: Excrypt Manager needs to be installed on the workstation that is being used to configure the HSM.

Excrypt Manager is a Windows application that can be used to configure the HSM in subsequent sections. HSM configuration can also be completed using FXCLI, the Excrypt Touch, or the Guardian Series 3. For more information about using these tools/devices to configure the HSM, please see the relevant Administrator's Guide.

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.

- Run the Excrypt Manager installer as an administrator.


Excrypt Manager Setup

Welcome to Excrypt Manager Setup

Welcome to the Excrypt Manager Setup Wizard.

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

The installation wizard will ask you to specify where you want Excrypt Manager to be installed. The default location is "*C:\Program Files\Futurex\Excrypt Manager*". Once that is done click "Install".

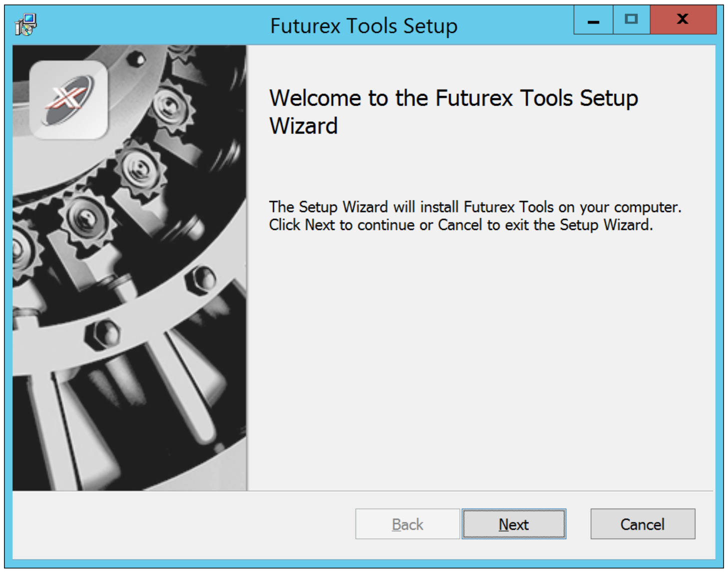
[5] INSTALL FUTUREX COMMAND LINE INTERFACE (FXCLI)

NOTE: FXCLI needs to be installed on the workstation that is being used to configure the HSM.

[5.1] INSTRUCTIONS FOR INSTALLING FXCLI IN WINDOWS

As mentioned in section 4, **Futurex Client Tools (FXCLI)** is included in the **FXTools** installation package. Just as with the **Futurex PKCS #11 (FXPKCS11)** module, the easiest way to install FXCLI on Windows is through installing FXTools. FXTools can be downloaded from the Futurex Portal.

- Run the FXTools installer as an administrator



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

By default, all tools are installed on the system. A user can overwrite and choose not to install certain modules.

NOTE: Since FXTools is only being used to install FXCLI in this case, it is not necessary to include any of the other services in the installation.

- **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 –**The client used to connect a Futurex Excrypt Touch to a local laptop, via USB, and a remote Futurex device.

[5.2] INSTRUCTIONS FOR INSTALLING FXCLI IN LINUX

Download the FXCLI module

Users can download the appropriate FXCLI package files for their system from the Futurex Portal.

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

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

If installing an `.rpm` package, run the following command in a terminal:

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

If installing a `.deb` package, run the following command in a terminal:

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

After the installation is completed, system environment variables must be defined for the location of the FXCLI binaries. To do so permanently you must add the following two lines to your *.bashrc* file:

```
PATH=$PATH:/usr/bin/fxcli-hsm
PATH=$PATH:/usr/bin/fxcli-kmes
```

[6] INSTALL FXJCE FILES

The Java provider relies on a JNI (Java Native Interface) library, which must be in the server's *$JAVA_ HOME/jre/lib* directory. It also requires a provider, which should be saved in the *$JAVA_HOME/jre/lib/ext* directory.

Extract the files from the zip file (*fxjce-OperatingSystem_x.xx.zip*) corresponding to the operating system in the working folder. Examples for each operating system are below:

Linux:

libfxjp11.so (library) -> *$JAVA_HOME/jre/lib/ext*

sunpkcs11-fx.jar (extension) -> *$JAVA_HOME/jre/lib/ext*

Windows:

fxjp11.dll (library) -> *C:\Program Files\Java\jre\lib\ext*

sunpkcs11-fx.jar (extension) -> *C:\Program Files\Java\jre\lib\ext*

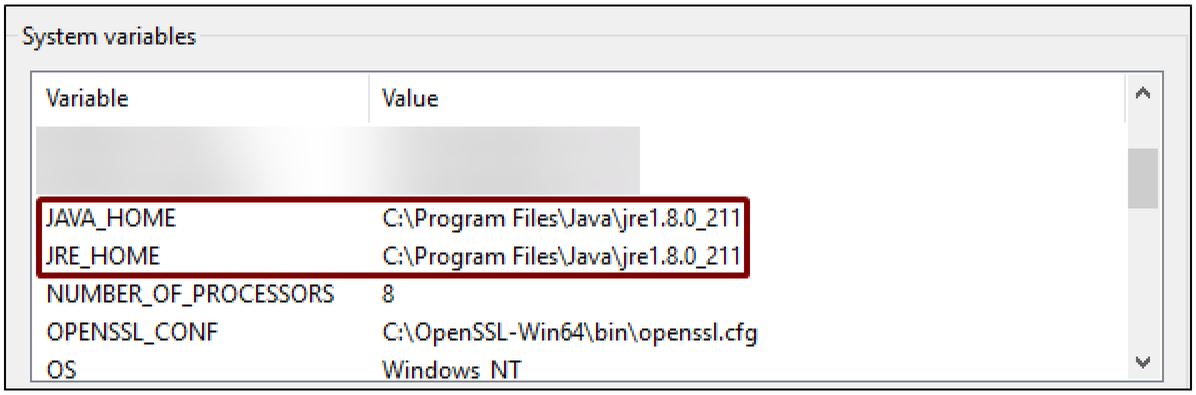
[7] SETTING SYSTEM ENVIRONMENT VARIABLES FOR THE JAVA LIBRARY

System environment variables must be defined for the location of the Java library. The variable settings are:

- JAVA_HOME = path to Java directory
- JRE_HOME = path to Java directory
- PATH = ; (add all the paths described above)

Windows example:

- JAVA_HOME = C:\Program Files\Java\jre1.8.0_211
- JRE_HOME = C:\Program Files\Java\jre1.8.0_211
- PATH = ...; C:\Program Files\Java\jre1.8.0_211; C:\Program Files\Java\jre1.8.0_211\bin;



Variable	Value
JAVA_HOME	C:\Program Files\Java\jre1.8.0_211
JRE_HOME	C:\Program Files\Java\jre1.8.0_211
NUMBER_OF_PROCESSORS	8
OPENSSL_CONF	C:\OpenSSL-Win64\bin\openssl.cfg
OS	Windows_NT

FIGURE: SAMPLE SYSTEM VARIABLE SETTINGS

Linux example:

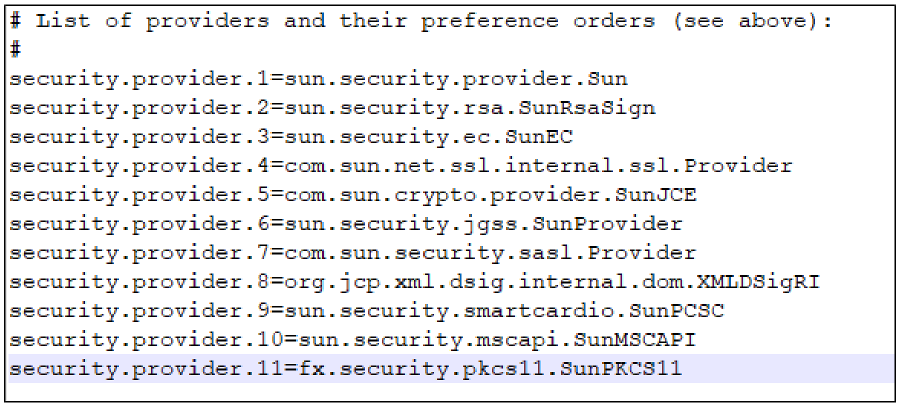
To define a system environment variable for the location of the Java library in Linux, you need to add the following line to your **.bashrc file:** (**NOTE**: This may be slightly different depending on where the Java library is installed on your system.)

```
PATH=$PATH:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/ext
```

[8] REGISTERING THE JAVA PROVIDER

The Java provider must be registered before it can be used. To register, modify the *java.security* file on the system (typically located in *$JAVA_HOME/jre/lib/security*). Append a line similar to the following to the provider list in the *java.security* file:

```
security.provider.11=fx.security.pkcs11.SunPKCS11
```



```
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=sun.security.ec.SunEC
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=sun.security.mscapi.SunMSCAPI
security.provider.11=fx.security.pkcs11.SunPKCS11
```

FIGURE: SAMPLE JAVA.SECURITY FILE

[9] CONFIGURE THE FUTUREX HSM

In order to establish a connection between the PKCS #11 library and the Futurex HSM, a few configuration items need to first be performed, which are the following:

NOTE: All of the steps in this section can be completed through either Excrypt Manager or FXCLI (if using a physical HSM rather than a virtual HSM). Optionally, steps 4 through 6 can be completed through the Guardian Series 3, which will be covered in Appendix A.

1. Connect to the HSM via the front USB port (**NOTE**: 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

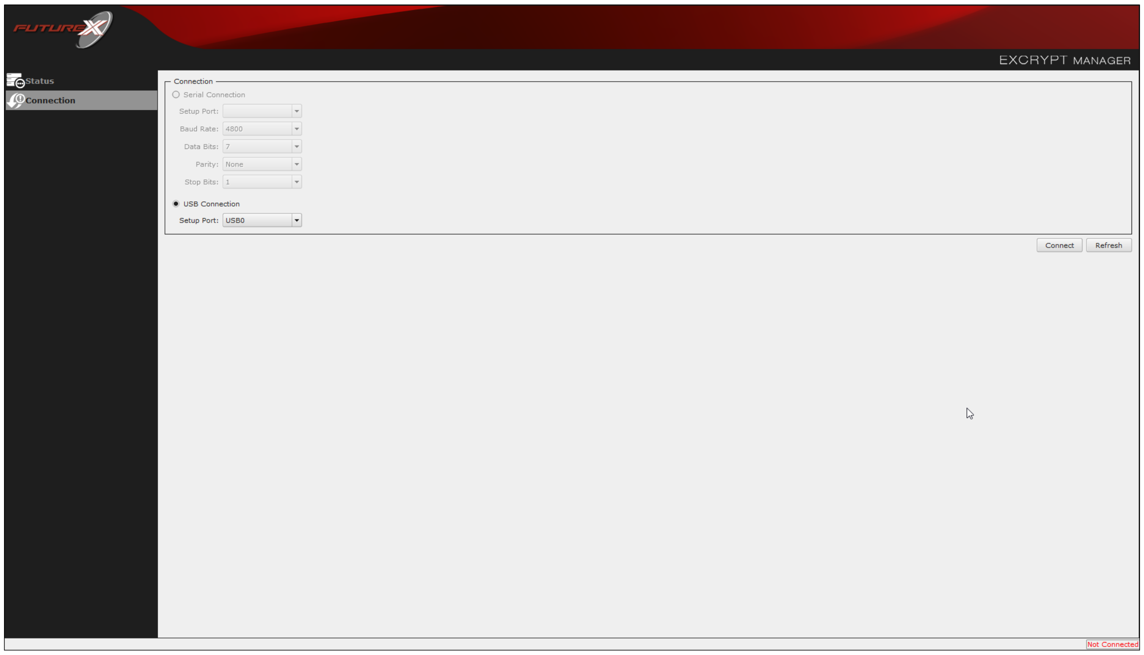
Each of these action items is detailed in the following subsections.

[9.1] 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.

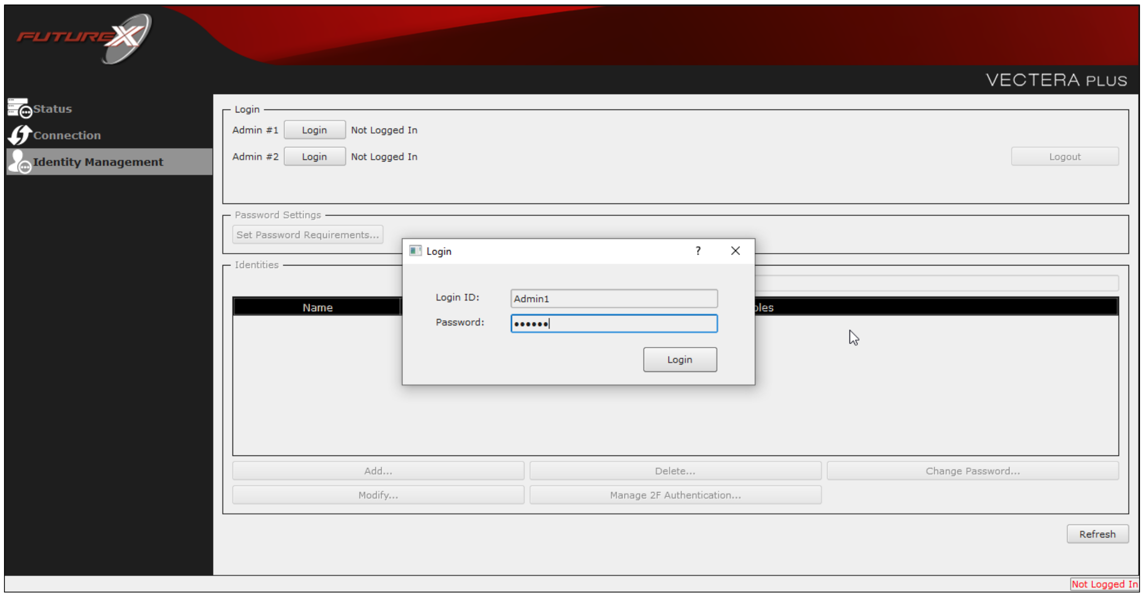
Connecting via Excrypt Manager

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



Connection
Serial Connection
Setup Port:
Baud Rate: 4800
Data Bits: 7
Parity: None
Stop Bits: 1
[✓ USB Connection]
Setup Port: USB0

Login with both default Admin identities.

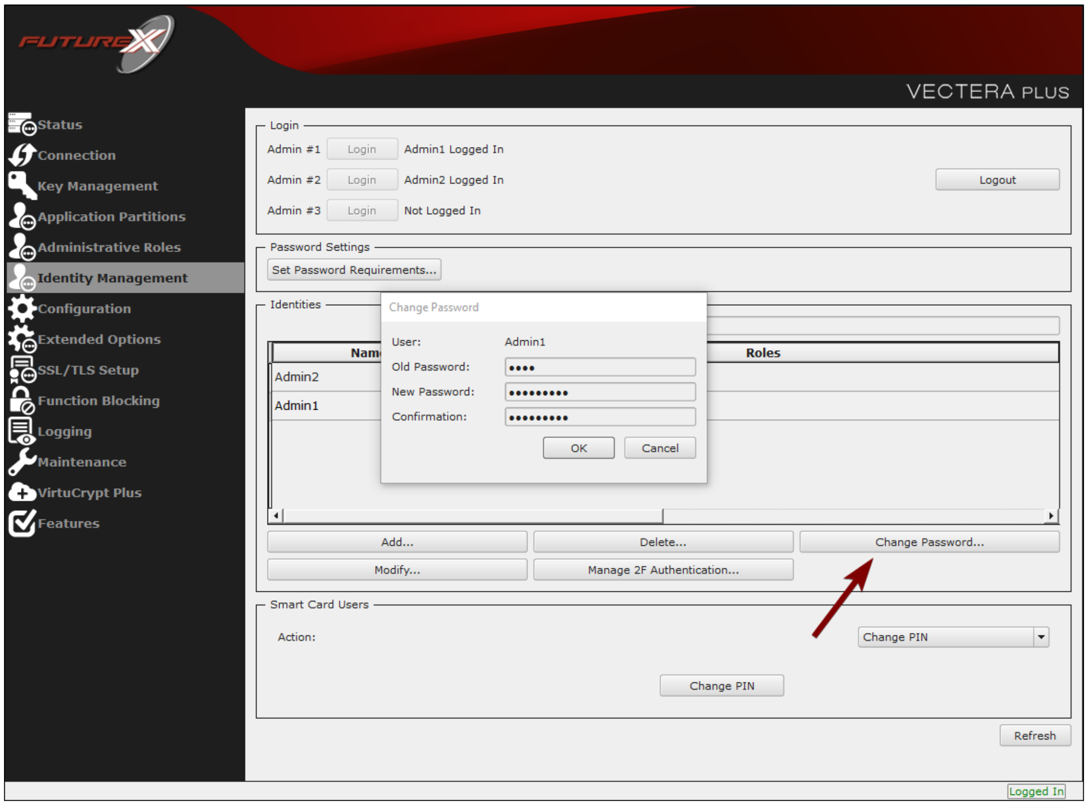


Login
Admin #1 Login Not Logged In
Admin #2 Login Not Logged In

Login
Login ID: Admin1
Password: ••••••
Login

The default Admin passwords (i.e. “safe”) must be changed 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 navigate to the Identity Management menu, select the first default Admin identity (e.g. “Admin1”), then click the “Change Password…” button. Enter the old password, then enter the new password twice, and click “OK”. Perform the same steps as above for the second default Admin identity (e.g. “Admin2”).



Admin #1 Login Admin1 Logged In
Admin #2 Login Admin2 Logged In
Admin #3 Login Not Logged In

Admin2
Admin1

Change Password
User: Admin1
Old Password: ••••
New Password: •••••••••
Confirmation: •••••••••

Action: Change PIN

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 will need to run it twice because you must login with both default Admin identities.

The default Admin passwords (i.e. “safe”) must be changed 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.

[9.2] 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)

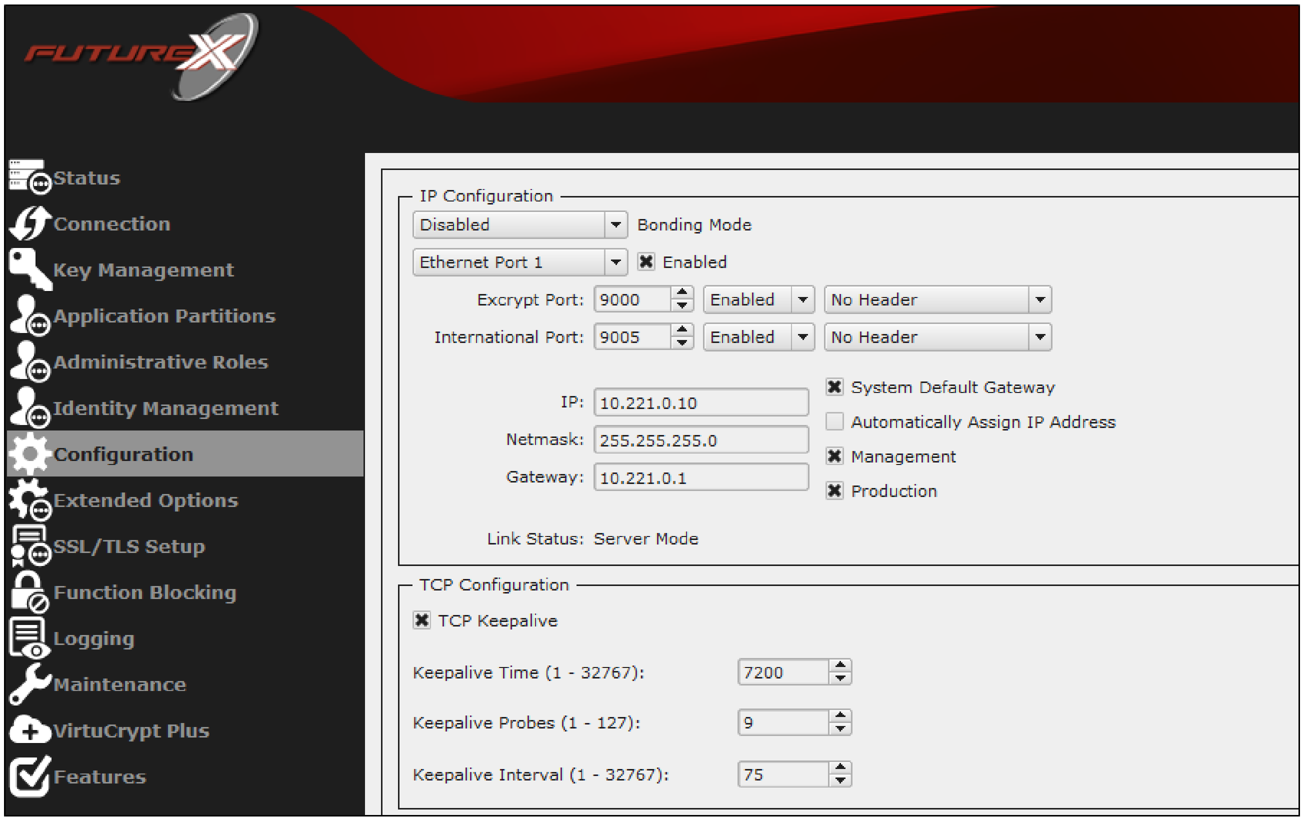
NOTE: For additional information about how to update features on your HSM, please refer to your HSM Administrator's Guide, section **"Download Feature Request File"**.

NOTE: Command Primary Mode = General Purpose, will enable the option to create the FTK major key in the HSM. This key will be required to be able 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.

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

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

Navigate to the *Configuration* page. There you will see the option to modify the IP configuration, as shown below:



IP Configuration

Disabled Bonding Mode

Ethernet Port 1 [✓ Enabled]

Excrypt Port: 9000 Enabled No Header

International Port: 9005 Enabled No Header

IP: 10.221.0.10

Netmask: 255.255.255.0

Gateway: 10.221.0.1

[✓ System Default Gateway]

Automatically Assign IP Address

[✓ Management]

[✓ Production]

Link Status: Server Mode

TCP Configuration

[✓ TCP Keepalive]

Keepalive Time (1 - 32767): 7200

Keepalive Probes (1 - 127): 9

Keepalive Interval (1 - 32767): 75

Alternatively, the following **FXCLI** command can be used 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:

- All of the remaining HSM configurations in this section can be completed 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, it may be necessary to synchronize the HSM after it is added to a Device Group on the Guardian.
- If configuration through a CLI is required for your use-case, then you should manage the HSMs directly.

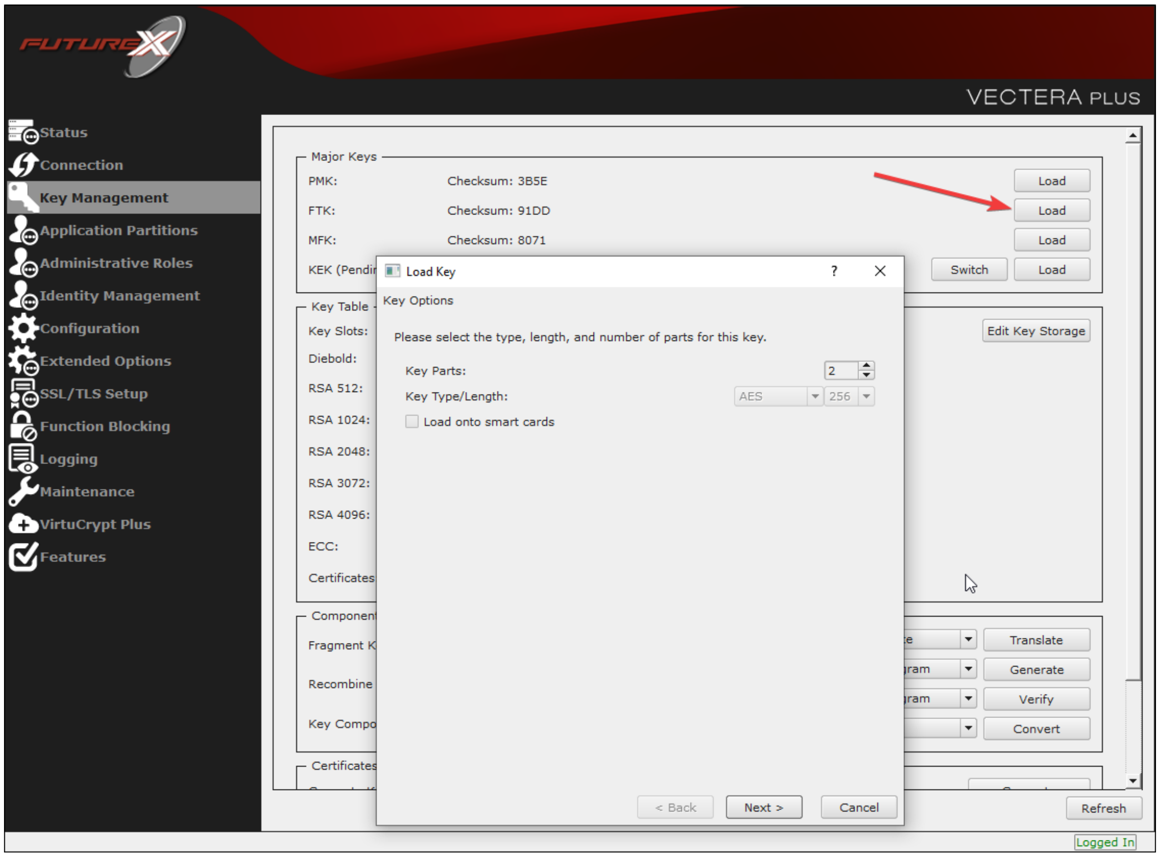
[9.4] LOAD FUTUREX KEY (FTK)

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

The FTK is used to wrap all keys stored on the HSM used with PKCS #11. If using multiple HSMs in a cluster, the same FTK can be used for syncing HSMs. Before an HSM can be used with PKCS #11, it must have an FTK.

NOTE: This process can also be completed 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*, then "Load" under FTK. Keys can be loaded as components that are XOR'd together, M-of-N fragments, or generated. If this is the first HSM in a cluster, it is recommended to generate the key and save to smart cards as M-of-N fragments.



Major Keys
PMK: Checksum: 3B5E
FTK: Checksum: 91DD
MFK: Checksum: 8071
KEK (Pendi

Key Table
Key Slots:
Diebold:
RSA 512:
RSA 1024:
RSA 2048:
RSA 3072:
RSA 4096:
ECC:
Certificates

Load Key
Key Options
Please select the type, length, and number of parts for this key.
Key Parts: 2
Key Type/Length: AES 256
Load onto smart cards

Alternatively, the following **FXCLI** commands can be used to load an FTK onto an HSM.

If this is the first HSM you are setting up you will need to generate a random FTK. Optionally, you can also load it 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 you will load the FTK from smart cards with the following command:

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

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

*For this step you will 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**. The default Administrator role and Admin identities can be used.*

NOTE: For the purposes of this integration guide you can consider the terms "Application Partition" and "Role" to be synonymous. For more information regarding Application Partitions, Roles, and Identities, please refer to the relevant Administrator's guide.

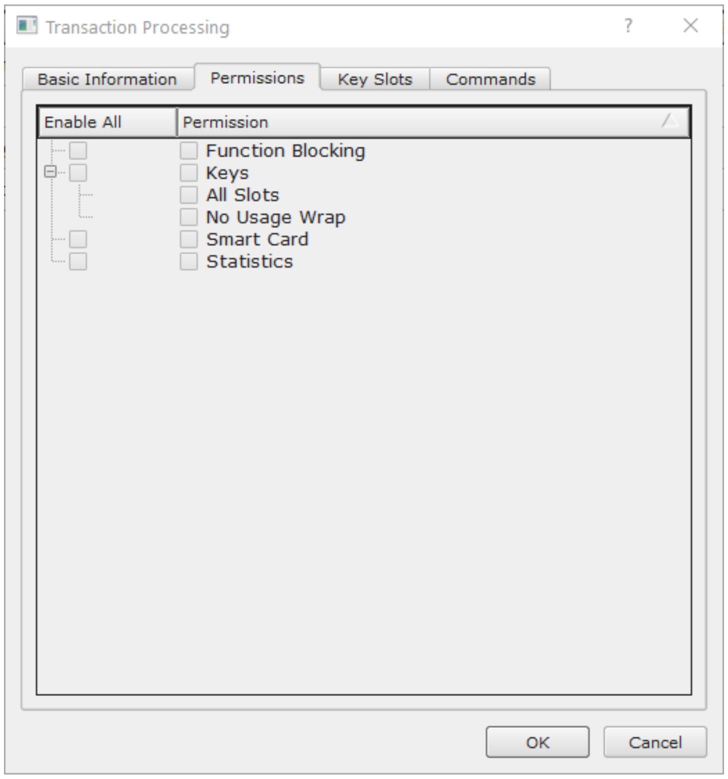
Configure a Transaction Processing Connection

Before an application logs in to the HSM with an authenticated user, it first connects via a “Transaction Processing” connection to the **Transaction Processing** Application Partition. For this reason, it is necessary to 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.

Navigate to the "Permissions" tab and ensure that the “All Slots” key permission is unchecked. None of the other key permissions should be enabled either.



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.

Lastly, 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:

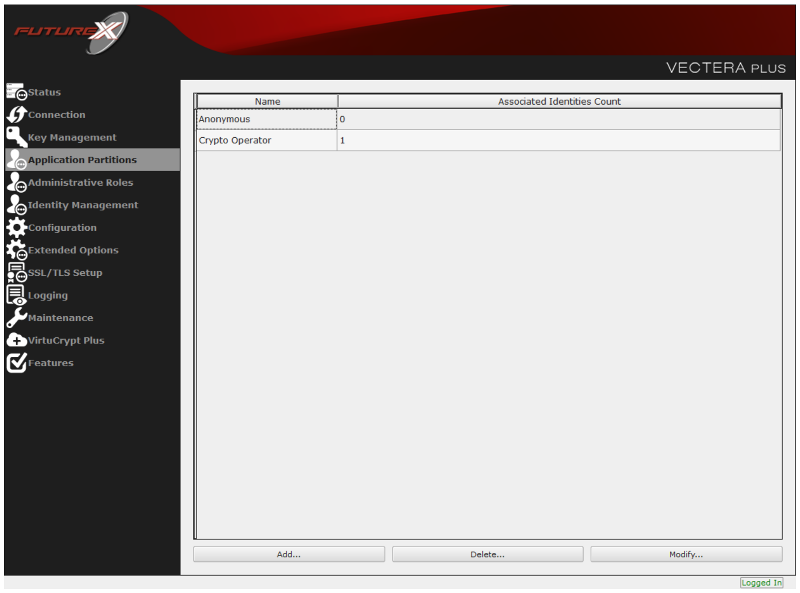
```
$ role modify --name Anonymous --clear-perms --clear-key-ranges
```

```
$ role modify --name Anonymous --add-perm 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

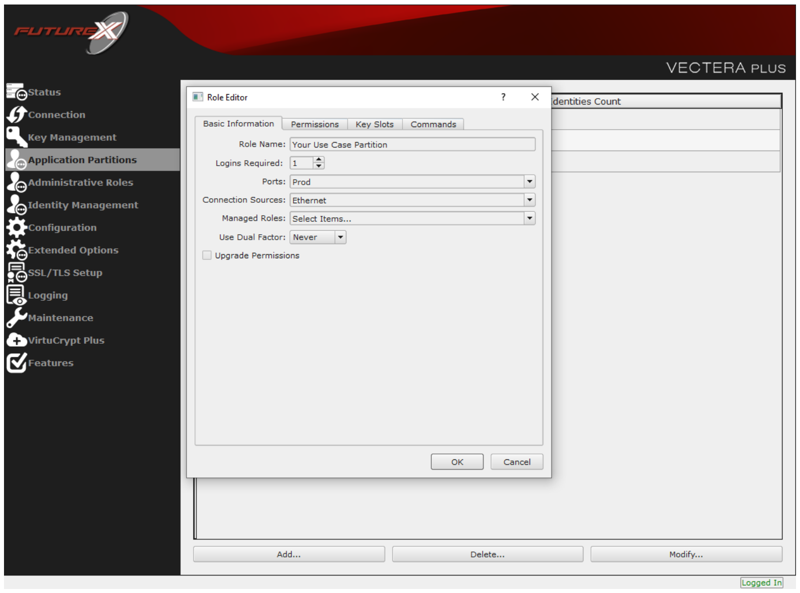
In order for application segregation to occur on the HSM, an Application Partition must be created specifically for your use case. Application partitions are used to segment the permissions and keys on an HSM between applications. The process for configuring a new application partition is outlined in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

Navigate to the *Application Partitions* page and click the "Add" button at the bottom.



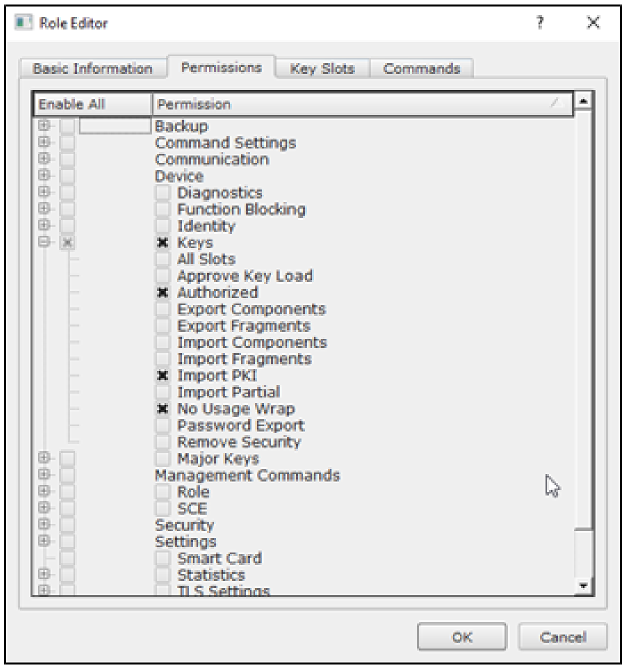
Name	Associated Identities Count
Anonymous	0
Crypto Operator	1

Fill in all of the fields in the *Basic Information* tab 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. *Logins Required* should be set to "1". *Ports* should be set to "Prod". *Connection Sources* should be configured to "Ethernet". The *Managed Roles* field should be left blank because we'll be specifying the exact Permissions, Key Slots, and Commands that we want this Application Partition/Role to have access to. Lastly, the *Use Dual Factor* field should be set to "Never".



Role Name: Your Use Case Partition
Logins Required: 1
Ports: Prod
Connection Sources: Ethernet
Managed Roles: Select Items...
Use Dual Factor: Never
Upgrade Permissions

Under the “Permissions” tab, select the key permissions shown in the screenshot below. 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 (This is only recommended if exchanging keys with external entities or using the HSM to wrap externally used keys).

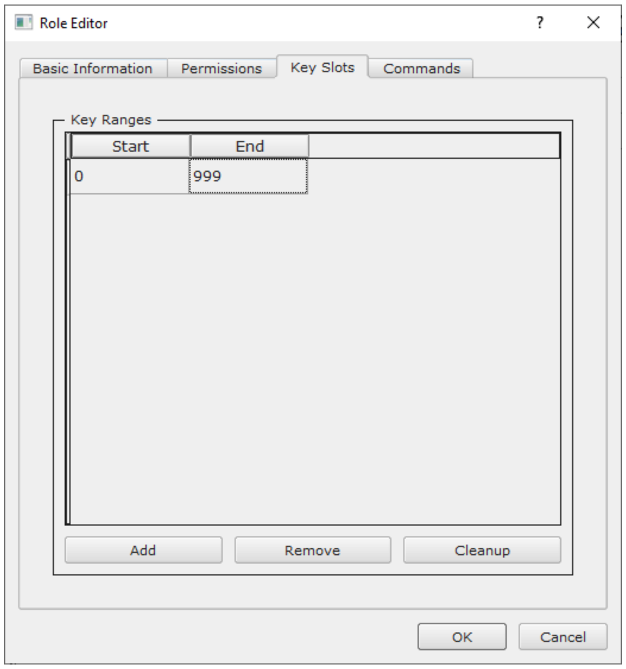


Backup
Command Settings
Communication
Device
- [ ] Diagnostics
- [ ] Function Blocking
- [ ] Identity
- [x] Keys
- [ ] All Slots
- [ ] Approve Key Load
- [x] Authorized
- [ ] Export Components
- [ ] Export Fragments
- [ ] Import Components
- [ ] Import Fragments
- [x] Import PKI
- [ ] Import Partial
- [x] No Usage Wrap
- [ ] Password Export
- [ ] Remove Security
- [ ] Major Keys

Management Commands
- [ ] Role
- [ ] SCE

Security
Settings
- [ ] Smart Card
- [ ] Statistics
- [ ] TLS Settings

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



Start	End
0	999

Based on application requirements there are particular functions that need to be enabled on the Application Partition in order to utilize the HSMs functionality. The most often used commands are included below. These can be enabled under the "Commands" tab.

PKCS #11 Communication Commands

- **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)

Key Operations Commands

- **APFP**: Generate PKI Public Key from Private Key
- **ASYL**: Load asymmetric key into key table
- **GECC**: Generate an ECC Key Pair
- **GPCA**: General purpose add certificate to key table
- **GPGS**: General purpose generate symmetric key
- **GPKA**: General purpose key add
- **GPKD**: General purpose key slot delete/clear
- **GRSA**: Generate RSA Private and Public Key
- **LRSA**: Load key into RSA Key Table
- **RPFP**: Get public components from RSA private key

Interoperable Key Wrapping

- **GPKU**: General purpose key unwrap (unrestricted)
- **GPUK**: General purpose key unwrap (preserves key usage)
- **GPKW**: General purpose key wrap (unrestricted)
- **GPWK**: General purpose key wrap (preserves key usage)

Data Encryption Commands

- **ADPK**: PKI Decrypt Trusted Public Key
- **GHSH**: Generate a Hash (Message Digest)
  *Starting in firmware version 7.x, this function is enabled by default and does not need to be specified.
- **GPED**: General purpose data encrypt and decrypt
- **GPGC**: General purpose generate cryptogram from key slot
- **GPMC**: General purpose MAC (Message Authentication Code)
- **GPSR**: General purpose RSA encrypt/decrypt or sign/verify with recovery
- **HMAC**: Generate a hash-based message authentication code
- **RDPK**: Get Clear Public Key from Cryptogram

Signing Commands

- **ASYS**: Generate a Signature Using a Private Key
- **ASYV**: Verify a Signature Using a Public Key
- **GPSV**: General purpose data sign and verify
- **RSAS**: Generate a Signature Using a Private Key

Alternatively, the following **FXCLI** commands can be used to create the new Application Partition and enable all of the functions that are needed:

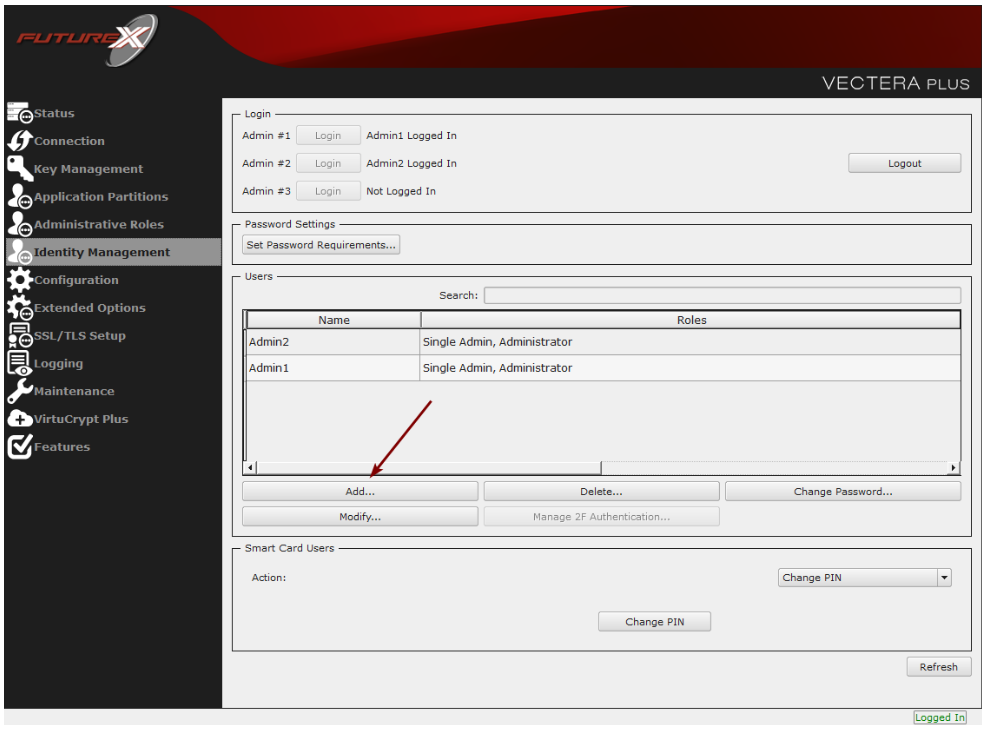
```
$ role add --name Role_Name --application --key-range (0,999) --perm "Keys:Authorized" --perm "Key-
s:Import PKI" --perm "Keys:No Usage Wrap"
```

```
$ role modify --name [role_name] --clear-perms --add-perm Excrypt:ECHO --add-perm Excrypt:PRMD --add-
perm Excrypt:RAND --add-perm Excrypt:HASH --add-perm Excrypt:GPKM --add-perm Excrypt:GPKS --add-perm
Excrypt:GPKR --add-perm Excrypt:APFP --add-perm Excrypt:ASYL --add-perm Excrypt:GECC --add-perm
Excrypt:GPCA --add-perm Excrypt:GPGS --add-perm Excrypt:GPKA --add-perm Excrypt:GPKD --add-perm
Excrypt:GRSA --add-perm Excrypt:LRSA --add-perm Excrypt:RPFP --add-perm Excrypt:GPKU --add-perm
Excrypt:GPUK --add-perm Excrypt:GPKW --add-perm Excrypt:GPWK --add-perm Excrypt:ADPK --add-perm
Excrypt:GHSH --add-perm Excrypt:GPED --add-perm Excrypt:GPGC --add-perm Excrypt:GPMC --add-perm
Excrypt:GPSR --add-perm Excrypt:HMAC --add-perm Excrypt:RDPK --add-perm Excrypt:ASYS --add-perm
Excrypt:ASYV --add-perm Excrypt:GPSV --add-perm Excrypt:RSAS
```

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

For this step you will need to be logged in with an identity that has a role with permissions ***Identity:Add****. The default Administrator role and Admin identities can be used.*

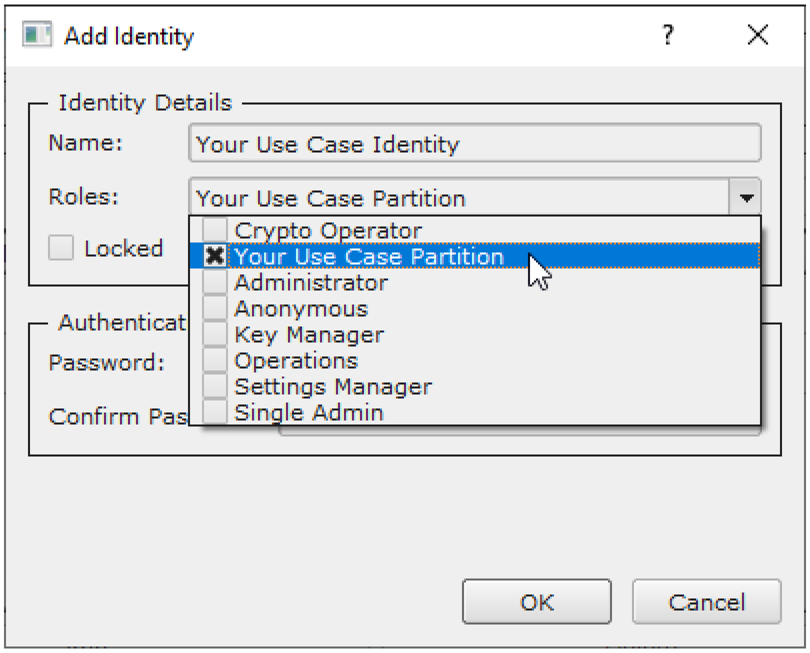
A new identity must be created, which will need to be associated with the Application Partition created in the previous step. To create this new identity, go to *Identity Management*, and click "Add".



Admin #1 Admin1 Logged In
Admin #2 Admin2 Logged In
Admin #3 Not Logged In

Name	Roles
Admin2	Single Admin, Administrator
Admin1	Single Admin, Administrator

Specify a name for the new identity, and in the Roles dropdown 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.


Add Identity

Identity Details

Name: Your Use Case Identity

Roles: Your Use Case Partition

Alternatively, the following **FXCLI** command can be used to create a new Identity and associate it with the role that was created:

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

This new identity must be set in 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>
```

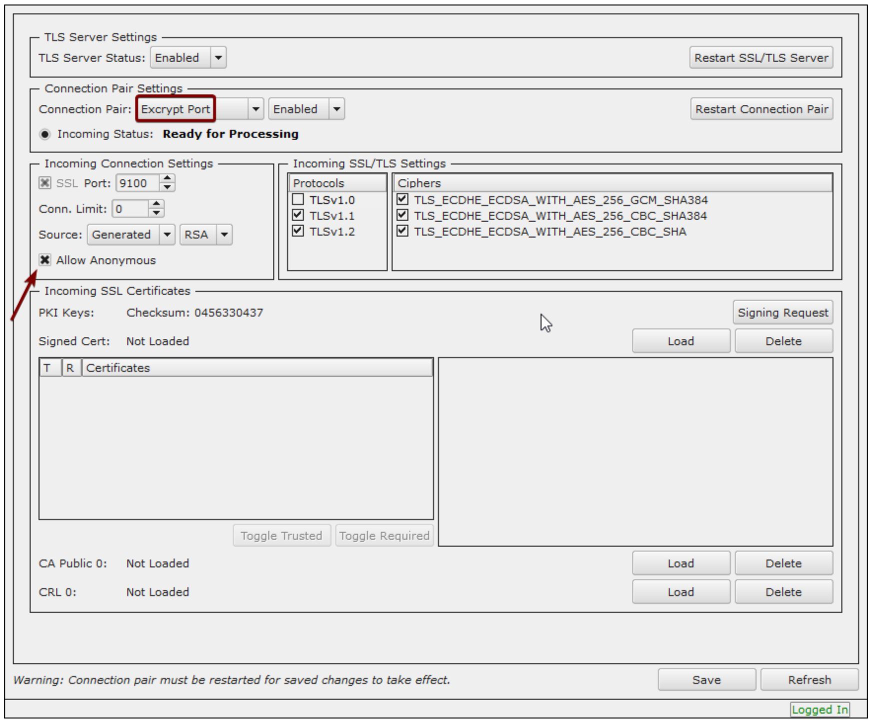
NOTE: Crypto Operator in the fxpkcs11.cfg file must match exactly the name of the identity created in the HSM.

[9.7] CONFIGURE TLS AUTHENTICATION

For this step you will 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****. The default Administrator role and Admin identities can be used.*

Enable Server-Side Authentication (Option 1)

Mutually authenticating to the HSM using client certificates is recommended, but server-side authentication is also supported. To enable server-side authentication go to *SSL/TLS Setup*, then select the Excrypt Port and enable the “Allow Anonymous” setting.


TLS Server Settings
TLS Server Status: Enabled
Connection Pair Settings
Connection Pair: Excrypt Port Enabled
Incoming Status: Ready for Processing
Incoming Connection Settings
[✓] SSL Port: 9100
Conn. Limit: 0
Source: Generated RSA
[✓] Allow Anonymous
Incoming SSL/TLS Settings

Protocols	Ciphers
☐ TLSv1.0	☑ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
☑ TLSv1.1	☑ TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
☑ TLSv1.2	☑ TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

Incoming SSL Certificates
PKI Keys: Checksum: 0456330437
Signed Cert: Not Loaded
CA Public 0: Not Loaded
CRL 0: Not Loaded
Warning: Connection pair must be restarted for saved changes to take effect.

Alternatively, the following **FXCLI** command can be used 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)

Mutually authenticating to the HSM using client certificates is recommended, and enforced by default. In the example below, FXCLI is utilized to generate a CA that then signs the HSM server certificate and a client certificate. The client keys and CSR are generated in Windows PowerShell with OpenSSL. For other options for managing certificates required for mutual authentication with the HSM, please review the relevant Administrator’s guide.

Find the **FXCLI** program that was installed with FXTools, and run it as an administrator.

Things to note:

- For this example, the computer running FXCLI is connected to the front port of the HSM. Remote management is possible however, using the HSMs Web Portal, or the Excrypt Touch.
- For commands that create an output file, if you do not specify a file path (as is the case here) it will save the file to the directory from which the FXCLI program is executed.
- 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 password.
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**: The following OpenSSL commands will need to be run 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.

```
## Make PKCS12 file
# Concatenate the signed client cert and private key into one pem file
$ cat SignedPki.pem >> Tree.pem
```

```
$ cat privatekey.pem >> Tree.pem
```

```
# 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 -in Tree.pem -out PKI.p12 -name "ClientPki" -password pass:safest
```

[10] EDIT THE FXPKCS11 CONFIGURATION FILE

The *fxpkcs11.cfg* file allows the user to set the PKCS #11 library to connect to the HSM. To edit, run a text editor as an Administrator and edit the configuration file accordingly. Most notably, the fields shown below must be set inside the **<HSM>** section (note that the full *fxpkcs11.cfg* file is not included).

NOTE: Our PKCS #11 library expects the PKCS #11 config file to be in a certain location (*C:\Program Files\Futurex\fxpkcs11\fxpkcs11.cfg* for Windows and */etc/fxpkcs11.cfg* for Linux), but that location can be overwritten using an environment variable (FXPKCS11_CFG).

```
# Connection information
<ADDRESS>             10.0.5.58           </ADDRESS>

# Load balancing
<FX-LOAD-BALANCE>           YES                </FX-LOAD-BALANCE>

# Log configuration
<LOG-FILE> C:\Program Files\Futurex\fxpkcs11\fxpkcs11.log </LOG-FILE>

# HSM crypto operator identity name
<CRYPTO-OPR>          [identity_name]     </CRYPTO-OPR>

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

# Production SSL information
<PROD-TLS-ANONYMOUS>    NO                </PROD-TLS-ANONYMOUS>
<PROD-TLS-CA>     C:\Program Files\Futurex\fxpkcs11\TlsCa.pem    </PROD-TLS-CA>
<PROD-TLS-CA>     C:\Program Files\Futurex\fxpkcs11\TlsProduction.pem    </PROD-TLS-CA>
<PROD-TLS-KEY>     C:\Program Files\Futurex\fxpkcs11\PKI.p12     </PROD-TLS-KEY>
<PROD-TLS-KEY-PASS>      safest          </PROD-TLS-KEY-PASS>
```

In the **<ADDRESS>** field, the IP of the HSM that the PKCS #11 library will connect to is specified.

If a Guardian is being used to manage HSMs in a cluster, the **<FX-LOAD-BALANCE>** field must be defined as "YES". If a Guardian is not being used it should be set to "NO".

In the **<LOG-FILE>** field, set the path to the PKCS #11 log file.

In the **<CRYPTO-OPR>** field, the name of the identity created in step 7.6 needs to be specified.

The **<PROD-ENABLED>** and **<PROD-PORT>** fields declare that the PKCS #11 library will connect to Production port 9100.

The **<PROD-TLS-ANONYMOUS>** field defines whether the PKCS #11 library will be authenticating to the server or not.

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, it is not necessary to define the signed client cert with the **<PROD-TLS-CERT>** tag, or the CA cert/s with one or more instances of the **<PROD-TLS-CA>** tag.

For additional details reference the Futurex PKCS #11 technical reference found on the Futurex Portal.

Once the *fxpkcs11.cfg* is edited, run the *PKCS11Manager* file to test the connection against the HSM, and check the *fxpkcs11.log* for errors and information. For more information, see our Administrator's Guide.

[11] JAVA KEYSTORE CREATION

A server's secure connections rely on a private server key and certificate to be stored in the Java KeyStore and saved on the HSM. This server certificate will be presented to clients when connecting to the server. The KeyStore is typically created using the Keytool application bundled with Java (Typically located in *$JAVA_ HOME/jre/bin/*). The following steps outline the KeyStore creation process, once again using Apache Tomcat as an example application:

- Generate a server keypair (which also includes a self-signed certificate that will be stored in the HSM.)
- Generate and export a CSR (Certificate Signing Request) to be signed by an External CA (if needed.)
- Import the external CA root certificate and server certificate signed by the External CA.

NOTE: For testing purposes, meaning to verify that the self-signed certificate is being created in the HSM and that the server is presenting it to client connections, it is enough to execute only section 9.1, *Generate a Server Keypair and Self-signed Certificate*, and then move on to server configuration.

NOTE: If a connection using an external CA is required, only then is it necessary to go to section 9.2, *Generate and Export CSR*, then sign the CSR using an external CA authority (which could be created with OpenSSL) and finally proceed with section 9.3, *Import CA Root Certificate*, and section 9.4, *Import Server Certificate Signed by CA*. For a full example of external CA creation please refer to APPENDIX B.

[11.1] GENERATE A SERVER KEYPAIR AND SELF-SIGNED CERTIFICATE

```
keytool -genkeypair -keyalg RSA -keysize 2048 -alias tomcatdemo1 -keystore NONE -storetype PKCS11 -
providername "Futurex" -providerclass "fx.security.pkcs11.SunPKCS11"
```

NOTE: *-alias* is a field used to set a name to identify the key pair and certificate to be generated. It can be any name (example: *tomcatdemo1*), but the same name must then be used in server configuration.

Upon the execution of the previous instruction, the Keytool application will ask for information for the server certificate to be generated.

Enter the KeyStore password: (This password must be saved. It may be required later for web server configuration and certificate importing.)

1. What is your first and last name?

[Unknown]: www.example.com

2. What is the name of your organizational unit?

[Unknown]: Engineering

3. What is the name of your organization?

[Unknown]: Futurex

4. What is the name of your City or Locality?

[Unknown]: Bulverde

5. What is the name of your State or Province?

[Unknown]: TX

6. What is the two-letter country code for this unit?

[Unknown]: US

7. Is CN=www.example.com, OU=Engineering, O=Futurex, L=Bulverde, ST=TX, C=US correct?

[no]: yes

NOTE: Command [9.1] generated a self-signed certificate. If a CA-Signed certificate is required, continue with steps [9.2], [9.3], and [9.4] (see APPENDIX B for additional details.) If a CA-signed certificate is not required, proceed to server configuration.

[11.2] GENERATE AND EXPORT A CSR

```
keytool -certreq -alias tomcatdemo1 -file example.csr -keystore NONE -storetype PKCS11 -providername
"Futurex" -providerclass "fx.security.pkcs11.SunPKCS11"
```

Enter KeyStore password:

The CSR must be signed by a CA, either third-party or internal. Once signed, the server certificate returned by the CA will be imported along with the CA certificate.

[11.3] IMPORT A CA ROOT CERTIFICATE

```
keytool -import -trustcacerts -alias tomcatdemo_ca -keystore NONE -file ca.crt -storetype PKCS11 -pro-
vidername "Futurex" -providerclass "fx.security.pkcs11.SunPKCS11"
```

Enter KeyStore password:

1. Trust this certificate?

[no]: yes

2. Certificate was added to KeyStore

[11.4] IMPORT A SERVER CERTIFICATE (SERVER CERTIFICATE SIGNED BY CA)

```
keytool -importcert -alias tomcatdemo1 -keystore NONE -file server.crt -storetype PKCS11 -providername
"Futurex" -providerclass "fx.security.pkcs11.SunPKCS11"
```

Enter the KeyStore password:

Certificate reply was installed in KeyStore.

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 will help do 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 cannot be found 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

SOLUTIONS ARCHITECT

E-mail: solutions@futurex.com

XCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

Live Chat available at http://www.futurex.com