OPENSSL ENGINE

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 Futurex HSMs with OpenSSL Engine using PKCS #11 libraries. For additional questions related to your HSM, see the relevant administrator's guide.

[1.2] APPLICATION DESCRIPTION

From the main OpenSC - libp11 page on GitHub (https://github.com/OpenSC/libp11): "OpenSSL implements various cipher, digest, and signing features and it can consume and produce keys. However plenty of people think that these features should be implemented in separate hardware, like USB tokens, smart cards or hardware security modules. Therefore OpenSSL has an abstraction layer called "engine" which can delegate some of these features to different piece of software or hardware.

engine_pkcs11 tries to fit the PKCS#11 API within the engine API of OpenSSL. That is, it provides a gateway between PKCS#11 modules and the OpenSSL engine API. One has to register the engine with OpenSSL and one has to provide the path to the PKCS#11 module which should be gatewayed to. This can be done by editing the OpenSSL configuration file, by engine specific controls, or by using the p11-kit proxy module."

[1.3] 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:

- Linux

Other:

- OpenSSL

[3] INSTALL FUTUREX PKCS #11 (FXPKCS11)

In a Linux environment, to install the Futurex PKCS #11 (FXPKCS11) module you need to download a tarball of the FXPKCS11 binaries from the Futurex Portal. Step by step installation instructions are provided below:

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 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)

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

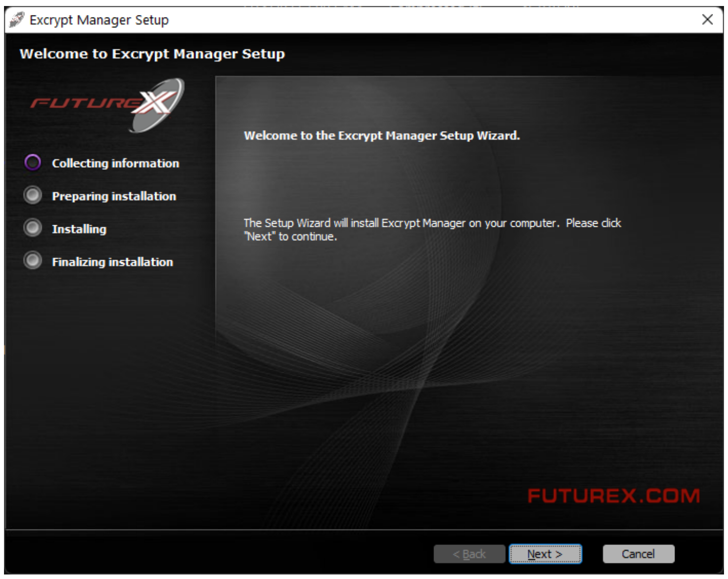
Note: If you will be configuring the Vectera Plus from a Linux computer, you can skip this section. If you will be configuring the Vectera Plus from a Windows computer, installing FXCLI in the next section is still required because FXCLI is the only method that can be used to configure TLS certificates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

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

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

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

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


Welcome to the Excrypt Manager Setup Wizard.

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

FIGURE: EXCRYPT MANAGER SETUP WIZARD

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

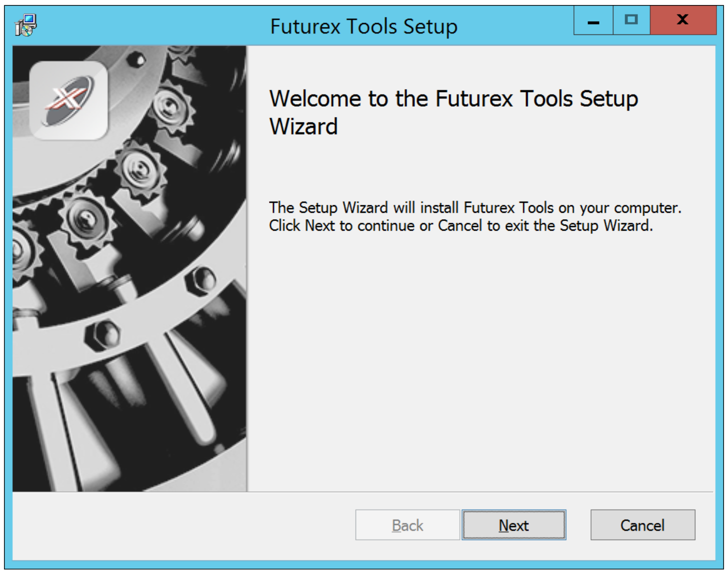
[5] INSTALL FUTUREX COMMAND LINE INTERFACE (FXCLI)

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

[5.1] INSTALLING FXCLI IN WINDOWS

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

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



Welcome to the Futurex Tools Setup Wizard

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

FIGURE: FUTUREX TOOLS SETUP WIZARD

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

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

[5.2] INSTALLING FXCLI IN LINUX

Download FXCLI

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

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

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

Futurex offers the following features for FXCLI:

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

Install FXCLI

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

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

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

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

Running FXCLI

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

```
$ fxcli-hsm
```

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

[6] CONFIGURE THE 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. Enable the EDSVWUX multi-usage combination for asymmetric keys
5. Load the Futurex FTK
6. Configure a Transaction Processing connection and create a new Application Partition
7. Create a new Identity that has access to the Application Partition created in the previous step
8. Configure TLS Authentication. There are two options for this:
    a. Enabling server-side authentication
    b. Creating client certificates for mutual authentication

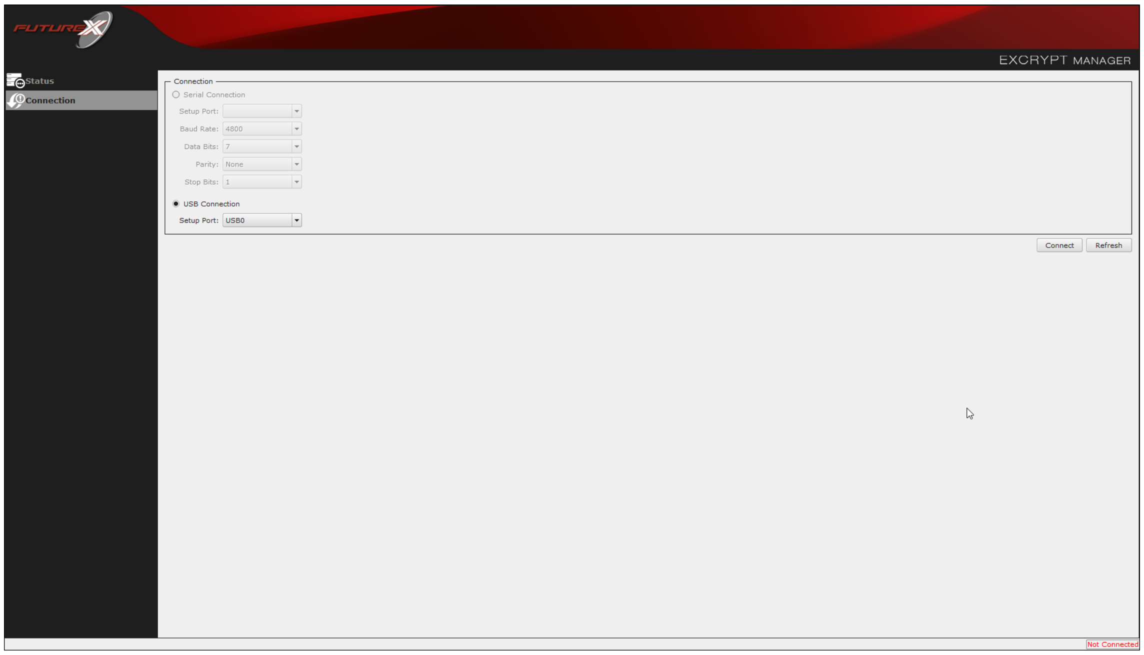
Each of these action items is detailed in the following subsections.

[6.1] 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

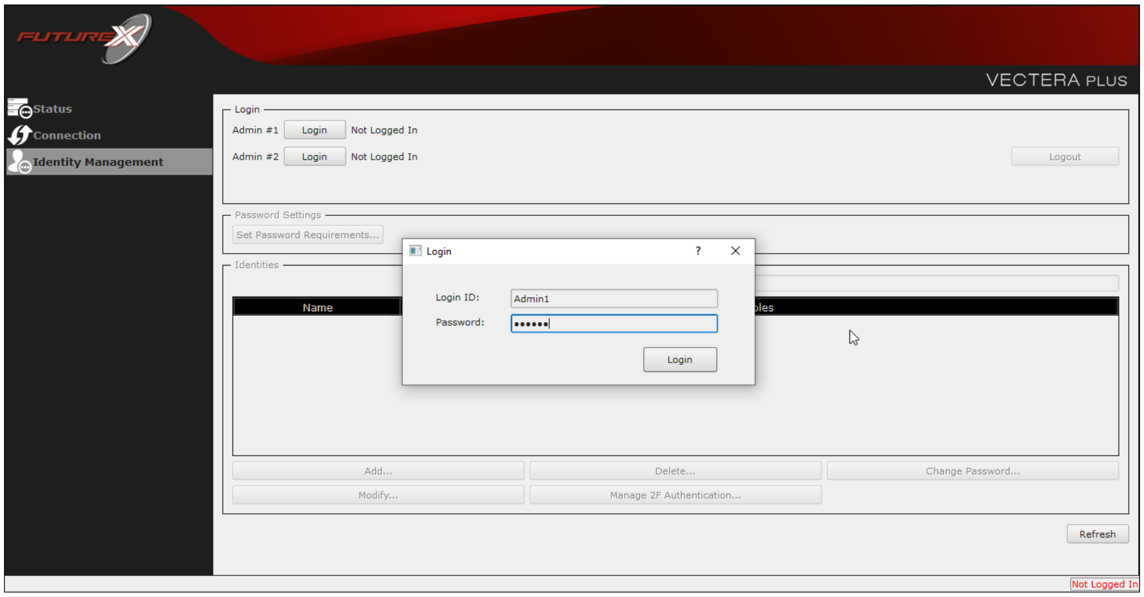
Open Excrypt Manager, click “Refresh” in the lower right-hand side of the Connection menu. Then select “USB Connection” and click “Connect”.



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

Not Connected

Login with both default Admin identities.



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

Password Settings

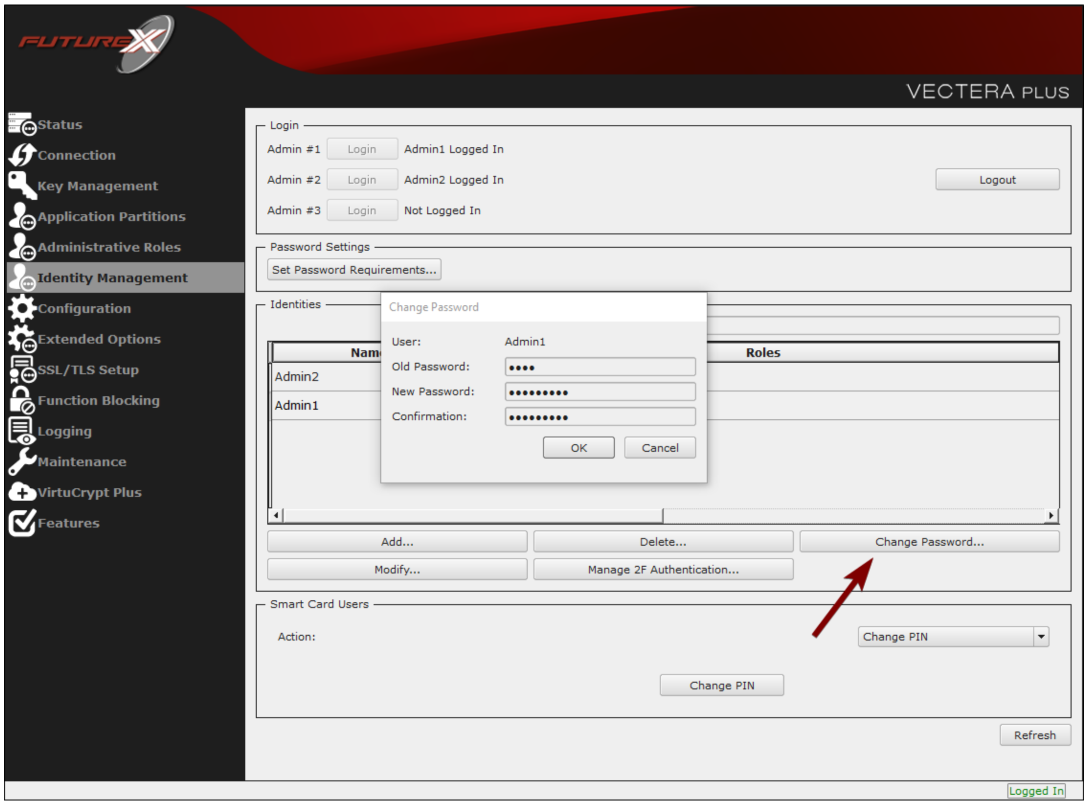
Identities
Name ...les

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

Not Logged In

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

Admin2
Admin1

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

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.

[6.2] 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)

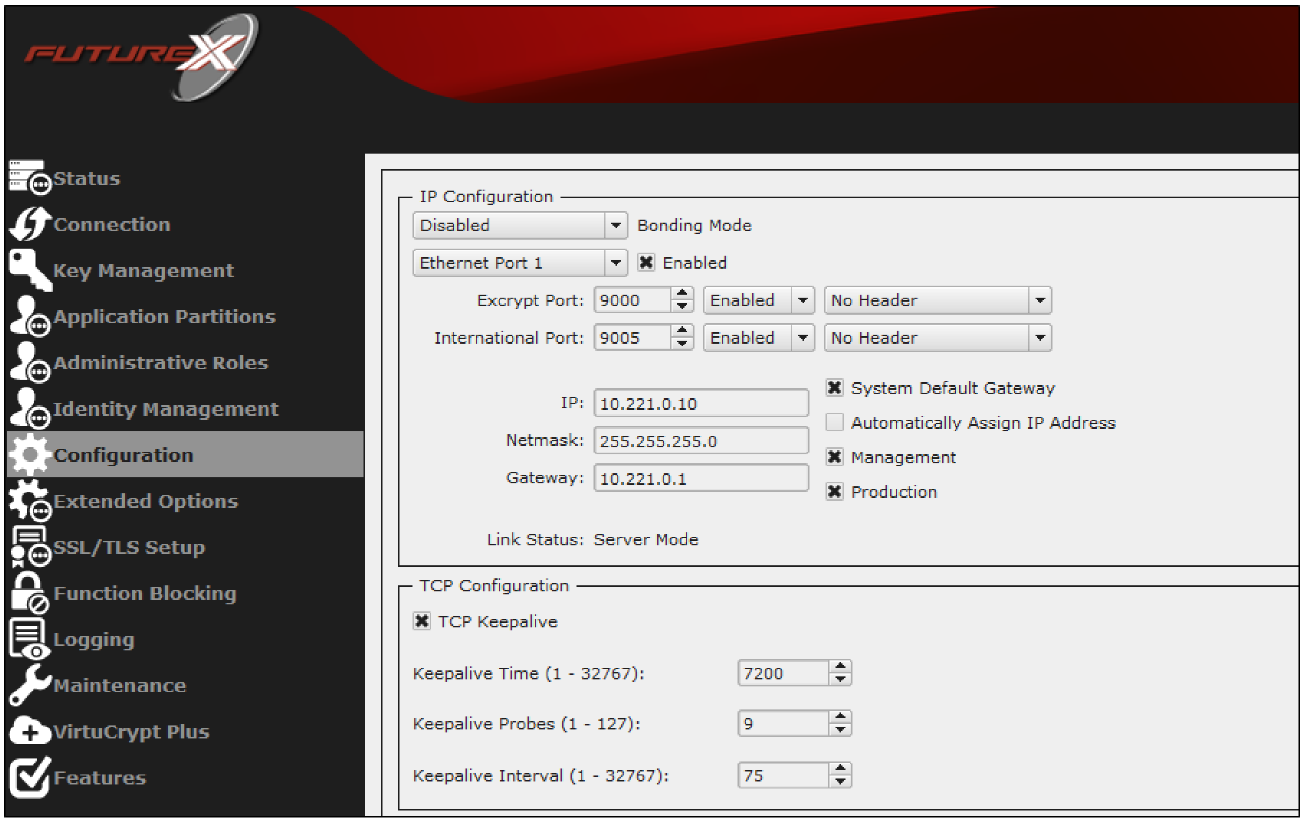
NOTE: For additional information about how to update features on your HSM, please refer to your HSM Administrator's Guide, section **"Download Feature Request File"**.

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.

[6.3] 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.

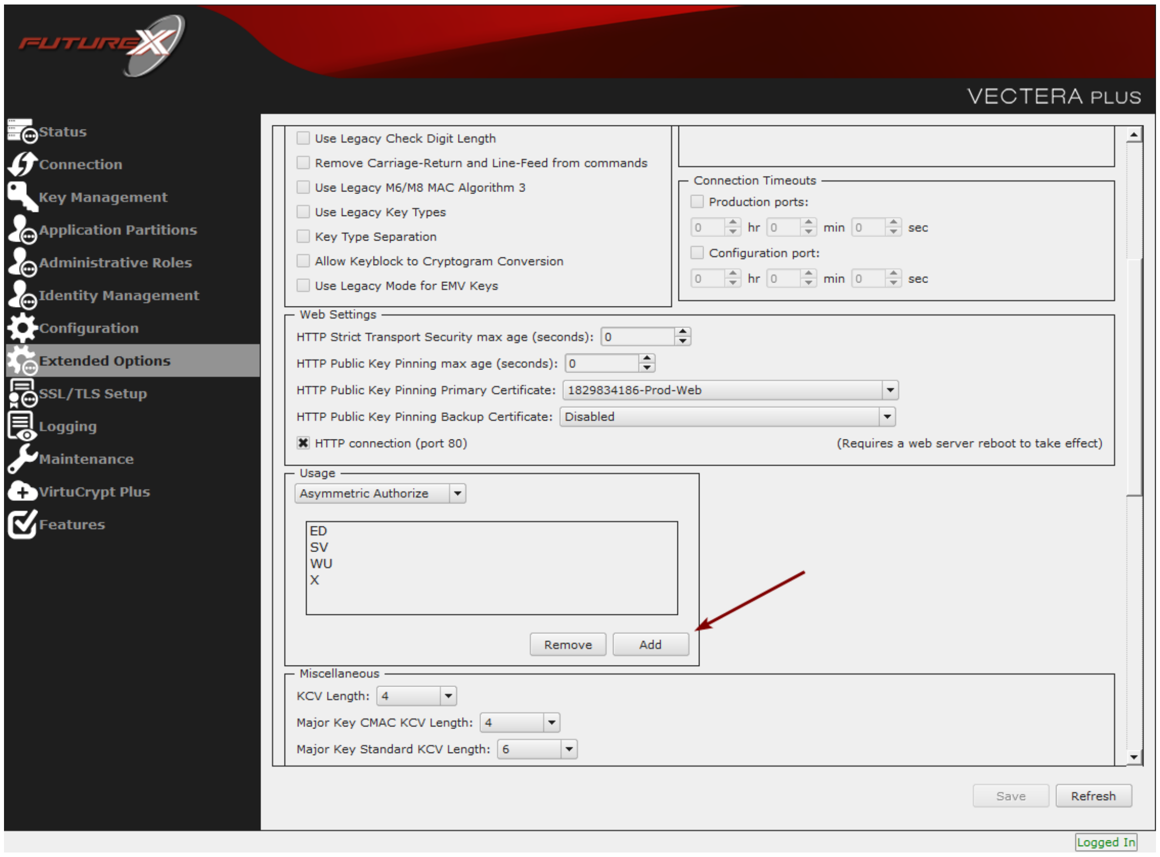
[6.4] ENABLE THE EDSVWUX MULTI-USAGE COMBINATION FOR ASYMMETRIC KEYS

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

Certain OpenSSL commands require asymmetric keys with multiple usages, which can be configured, but is not enabled by default on the Vectera Plus.

The specific multi-usage combination that is required is EDSVWUX (i.e., all usages). It is necessary to enable this multi-usage combination for all users, including anonymous users, due to how OpenSSL creates the keys on the HSM.

To configure this via Excrypt Manager, navigate to the *Extended Options* menu. In the "Usage" section, there is the option to add a new usage combination. With **Asymmetric Authorize** selected in the Usage dropdown, click the **Add** button.



- [ ] Use Legacy Check Digit Length
- [ ] Remove Carriage-Return and Line-Feed from commands
- [ ] Use Legacy M6/M8 MAC Algorithm 3
- [ ] Use Legacy Key Types
- [ ] Key Type Separation
- [ ] Allow Keyblock to Cryptogram Conversion
- [ ] Use Legacy Mode for EMV Keys

Connection Timeouts
- [ ] Production ports: 0 hr 0 min 0 sec
- [ ] Configuration port: 0 hr 0 min 0 sec

Web Settings
- HTTP Strict Transport Security max age (seconds): 0
- HTTP Public Key Pinning max age (seconds): 0
- HTTP Public Key Pinning Primary Certificate: 1829834186-Prod-Web
- HTTP Public Key Pinning Backup Certificate: Disabled
- [x] HTTP connection (port 80) (Requires a web server reboot to take effect)

Usage
- Asymmetric Authorize
- ED
- SV
- WU
- X

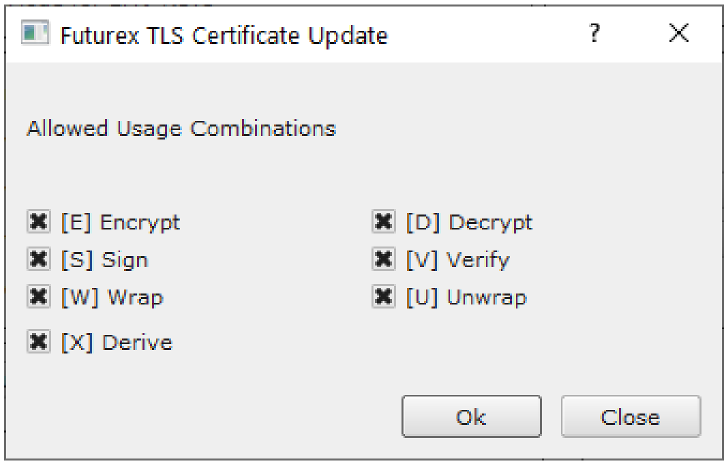
Remove Add

Miscellaneous
- KCV Length: 4
- Major Key CMAC KCV Length: 4
- Major Key Standard KCV Length: 6

Save Refresh

Logged In

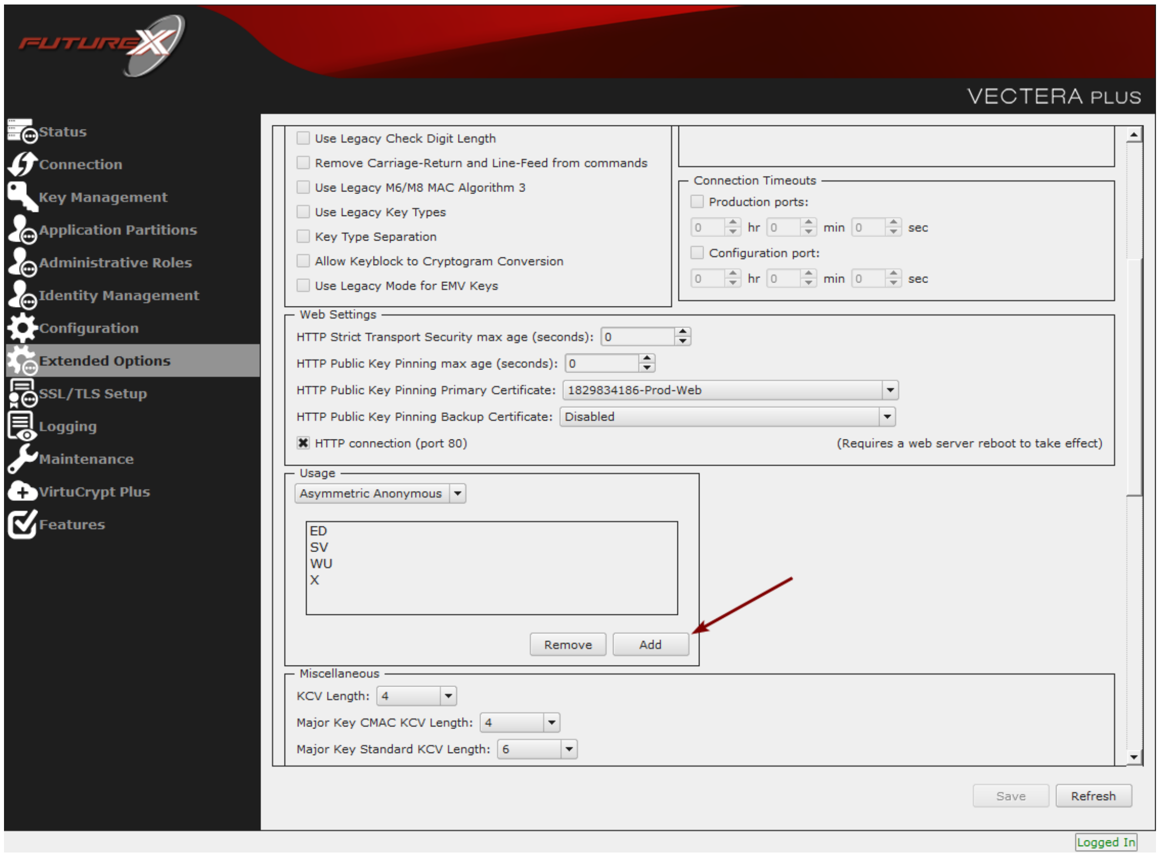
Select the EDSVWUX usage combination and click "Ok".


Futurex TLS Certificate Update

Allowed Usage Combinations

- [✓] [E] Encrypt
- [✓] [D] Decrypt
- [✓] [S] Sign
- [✓] [V] Verify
- [✓] [W] Wrap
- [✓] [U] Unwrap
- [✓] [X] Derive

Now select **Asymmetric Anonymous** in the Usage dropdown and enable the same EDSVWUX usage combination.



- [ ] Use Legacy Check Digit Length
- [ ] Remove Carriage-Return and Line-Feed from commands
- [ ] Use Legacy M6/M8 MAC Algorithm 3
- [ ] Use Legacy Key Types
- [ ] Key Type Separation
- [ ] Allow Keyblock to Cryptogram Conversion
- [ ] Use Legacy Mode for EMV Keys

Connection Timeouts
- [ ] Production ports: 0 hr 0 min 0 sec
- [ ] Configuration port: 0 hr 0 min 0 sec

Web Settings
HTTP Strict Transport Security max age (seconds): 0
HTTP Public Key Pinning max age (seconds): 0
HTTP Public Key Pinning Primary Certificate: 1829834186-Prod-Web
HTTP Public Key Pinning Backup Certificate: Disabled
- [✓] HTTP connection (port 80) (Requires a web server reboot to take effect)

Usage
Asymmetric Anonymous
ED
SV
WU
X

Miscellaneous
KCV Length: 4
Major Key CMAC KCV Length: 4
Major Key Standard KCV Length: 6

Click the "Save" button on the bottom-right-hand side of the window to save the changes.

Alternatively, the following **FXCLI** commands can be used to add the EDSVWUX multi-usage combination for asymmetric keys for all users:

```
$ multi-usage add --asymmetric --auth -e -d -s -v -w -u -x
```

```
$ multi-usage add --asymmetric --anon -e -d -s -v -w -u -x
```

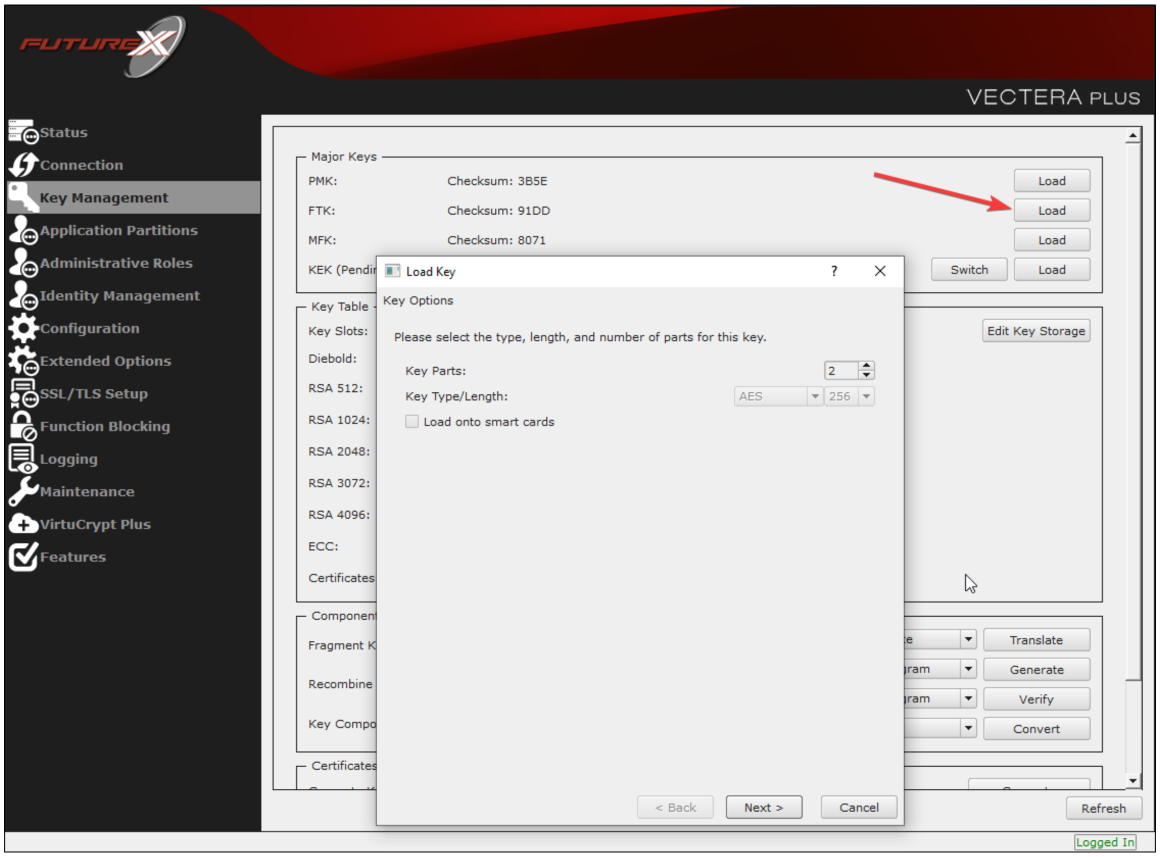
[6.5] LOAD FUTUREX KEY (FTK)

*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.



PMK: Checksum: 3B5E
FTK: Checksum: 91DD
MFK: Checksum: 8071

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

< Back Next > Cancel

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
```

[6.6] 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.

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.

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:

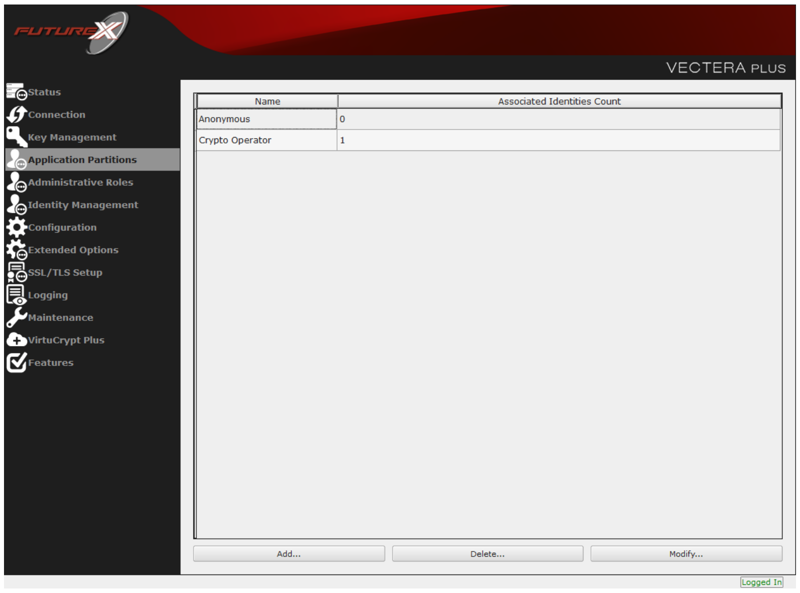
```
$ role modify --name Anonymous --clear-perms --clear-key-ranges
```

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

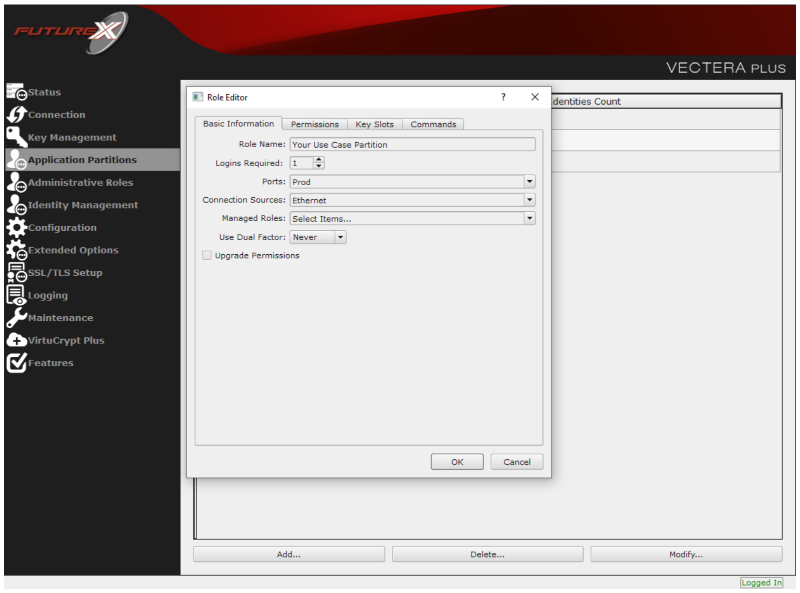
In order for application segregation to occur on the HSM, an Application Partition must be created specifically for your use case. Application partitions are used to segment the permissions and keys on an HSM between applications. The process for configuring a new application partition is outlined in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

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



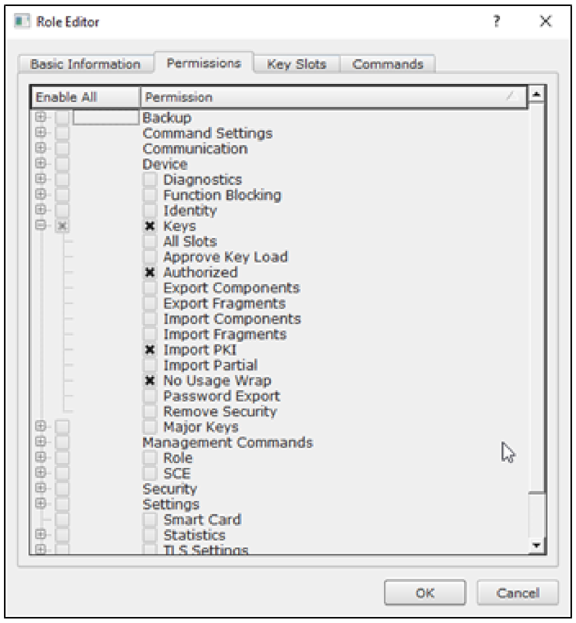
Name	Associated Identities Count
Anonymous	0
Crypto Operator	1

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).

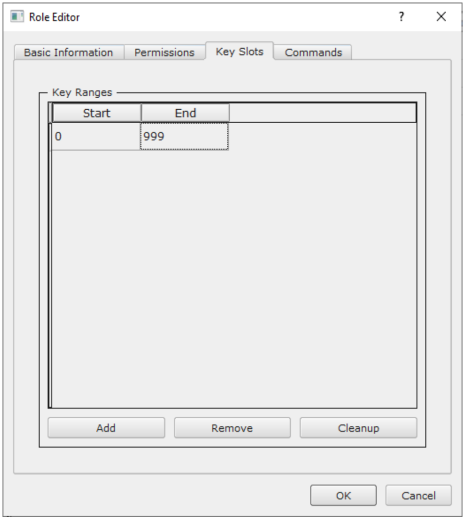


Backup
Command Settings
Communication
Device
- [ ] Diagnostics
- [ ] Function Blocking
- [ ] Identity
- [x] Keys
- [ ] All Slots
- [ ] Approve Key Load
- [x] Authorized
- [ ] Export Components
- [ ] Export Fragments
- [ ] Import Components
- [ ] Import Fragments
- [x] Import PKI
- [ ] Import Partial
- [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 commands that need to be enabled to perform the example OpenSSL commands in the last section of the integration guide are listed below. These can be enabled under the "Commands" tab.

NOTE: Additional commands may need to be enabled on the Application Partition depending on which OpenSSL commands are being utilized in conjunction with the the pkcs11 OpenSSL engine.

PKCS #11 Communication Commands

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

Key Operations Commands

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

Data Encryption Commands

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

Miscellaneous Commands

- **TIME**: Set the HSM internal clock

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

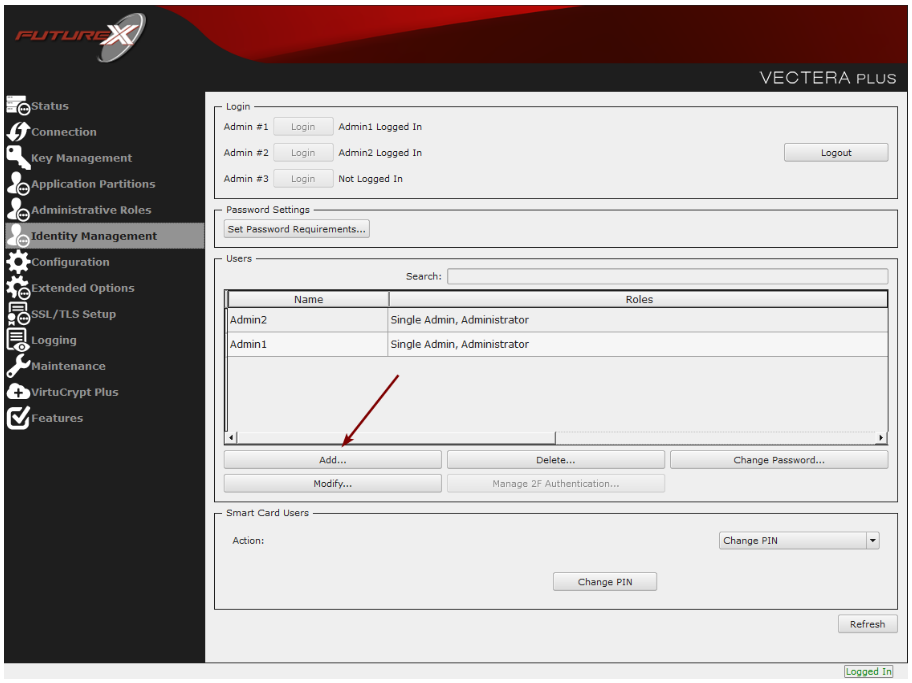
```
$ role add --name Role_Name --application --key-range (0,999) --perm "Keys:Authorized" --perm "Key-
s:Import PKI" --perm "Keys:No Usage Wrap"
```

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

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

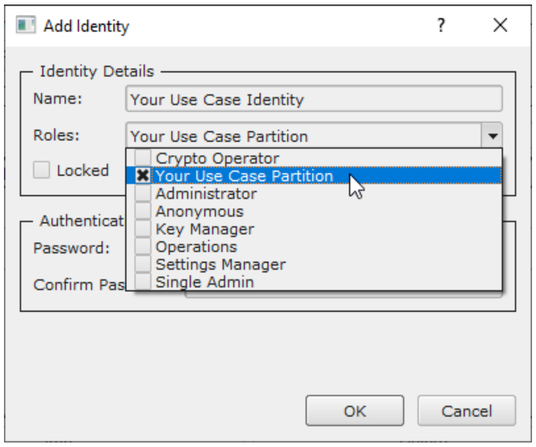
*For this step you will need to be logged in with an identity that has a role with permissions **Identity:Add**. The default Administrator role and Admin identities can be used.*

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”.



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.



Name: Your Use Case Identity
Roles: Your Use Case Partition
Locked

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>
```

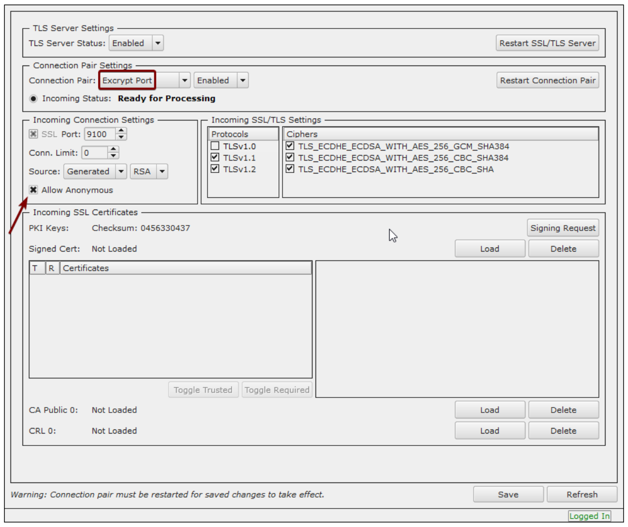
NOTE: Crypto Operator in the fxpkcs11.cfg file must match exactly the name of the identity created in the HSM.

[6.8] 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: TLSv1.0 [✓ TLSv1.1] [✓ TLSv1.2]
Ciphers: [✓ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384] [✓ TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384] [✓ 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.
Logged In

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 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*: 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.

```
# 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
```

[7] EDIT THE CONFIGURATION FILE

The `fxpkcs11.cfg` file allows the user to set the FXPKCS11 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: The Futurex PKCS #11 library expects the FXPKCS11 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`).

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

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

    # 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>
```

In the **<SLOT>** and **<LABEL>** fields we specify to use PKCS11 slot 0 and assign it the label "Futurex".

In the **<CRYPTO-OPR>** field, the name of the identity that was created for the application partition needs to be specified.

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

The **<PROD-PORT>** field declares 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.

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".

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, refer to the Futurex PKCS #11 technical reference found on the Futurex Portal.

[8] OPENSSL ENGINE INSTALLATION AND CONFIGURATION

This section will cover the installation and configuration of libp11, OpenSC, and the pkcs11 engine plugin for the OpenSSL library. An overview of these three libraries is provided below:

- libp11 provides a higher-level (compared to the PKCS#11 library) interface to access PKCS#11 objects. It is designed to integrate with applications that use OpenSSL.
- OpenSC provides a set of libraries and utilities to work with smart cards. Its main focus is on cards that support cryptographic operations, and facilitate their use in security applications such as authentication, mail encryption and digital signatures.
- pkcs11 engine plugin for the OpenSSL library allows accessing PKCS#11 modules in a semi-transparent way.

[8.1] INSTALL LIBP11 AND OPENSC

1. In a terminal, run the following sequence of commands to install libp11 and OpenSC:

```
sudo apt update
sudo apt install libengine-pkcs11-openssl
sudo apt install opensc
```

[8.2] EDIT THE OPENSSL CONFIGURATION FILE

1. Run the following command to determine the location of the OpenSSL configuration file for the logged in user:

```
openssl version -d
```

   NOTE: If you prefer to edit your global OpenSSL configuration file, it's location is most often in */etc/ssl/openssl.cnf*.

2. Open in a text editor the *openssl.cnf* file that is inside of the OpenSSL directory determined from the previous command.

   This line must be placed at the top, before any sections are defined:

```
openssl_conf = openssl_init
```

   This needs be added to the bottom of the file:

```
[openssl_init]
engines=engine_section
[engine_section]
pkcs11 = pkcs11_section
[pkcs11_section]
engine_id = pkcs11
dynamic_path = /usr/lib/x86_64-linux-gnu/engines-1.1/pkcs11.so
MODULE_PATH = /usr/local/bin/fxpkcs11/libfxpkcs11.so
PIN = "safest"
init = 1
```

NOTE: The value set for MODULE_PATH needs to be specific to where the Futurex PKCS #11 module is installed on your system.

NOTE: The password of the identity created on the Vectera Plus needs to be set in the PIN field.

[9] TESTING OPENSSL ENGINE

[9.1] SET FXPKCS11 ENVIRONMENT VARIABLES

In a terminal, run the following sequence of commands to set the required FXPKCS11 environment variables:

```
export FXPKCS11_MODULE=/path/to/libfxpkcs11.so;
export FXPKCS11_CFG=/path/to/fxpkcs11.cfg;
```

[9.2] CREATE A KEY PAIR ON THE VECTERA PLUS USING PKCS11-TOOL

In a terminal, run the following command to create a new key pair on the Vectera Plus using pkcs11-tool:

```
pkcs11-tool --module $FXPKCS11_MODULE --login --keypairgen --key-type rsa:2048 --label "my_rsa2048_
key" --id "123456" --usage-sign --usage-decrypt
```

Enter the password of the identity configured in the *fxpkcs11.cfg* file when prompted for the User PIN. If the command is successful the keys will be listed in the output, as shown below:

```
Key pair generated:
Private Key Object; RSA
  label:      my_rsa2048_key
  ID:         123456
  Usage:      decrypt, sign, unwrap
  Access:     sensitive, local
Public Key Object; RSA 2048 bits
  label:      my_rsa2048_key
  ID:         123456
  Usage:      encrypt, verify, wrap
  Access:     local
```

One private RSA 2048 key was created with asymmetric sign & verify usage, and one public RSA 2048 key was created with verify usage. These keys will be used in the test OpenSSL commands in the next section.

[9.3] OPENSSL EXAMPLE COMMANDS

Below are several OpenSSL example commands, most of which use the keys created on the Vectera Plus in the previous section. In all of the commands that utilize the keys created on the HSM, the pkcs11 OpenSSL engine is specified.

NOTE: The purpose of this section is not to provide an exhaustive list of OpenSSL commands that can be run using the pkcs11 OpenSSL Engine, but rather to give a few examples of use-cases and confirm that everything is configured correctly. Please refer to OpenSSL's documentation for the full list of compatible commands.

Example 1: Output the public key from the HSM

In a terminal, run the following command to output the public key from the HSM:

```
openssl rsa -engine pkcs11 -pubout -inform engine -in "pkcs11:object=my_rsa2048_key"
```

If the command is successful it should output the public key to screen, similar to what is shown below:

```
engine "pkcs11" set.
writing RSA key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoqFl+qYGJ9ou+tycLDCm
7RSTKxYcytiqA2yD3WGfrd72X8iAkuB2QL/IF/Kande1gSRaCTs5vnC0JZ9SP0nU
J3bY9b0GfXKR5kJsQGdQOKs29m0kyHjge7QRT6rfZuHhj8TRfqpPNzNnZU9MflMx
85XlTLE2HUV+e1vKHfkFC1gQrULDQ1ROb8HZKe13k7SIv4iMOZrswq7qgvyFFWOV
3Kn27yNsAKORMAoEPEwc5hre3rwJrP/W9I+EfFPDtMzI7wWPaQork3AE+bV3c8Dd
+Iv7fnXKPjK/n+4ctjnMfeTT/tG99ShkhkJkHRqGr4VNFv34hOQlwcJYr6NLrCA4
EQIDAQAB
-----END PUBLIC KEY-----
```

Example 2: Encrypt data with the public key and decrypt with the HSM stored private key

In a terminal, run the following command to generate a file called "clear_data" containing random ASII data:

```
echo "This is a test file" > ./clear_data
```

Retrieve the public key from the HSM.

```
openssl rsa -engine pkcs11 -inform ENGINE -in "pkcs11:object=my_rsa2048_key" -pubout -outform PEM -
out pubkey.pem
```

Encrypt the "clear_data" file using the public key retrieved from the HSM and output the results to a file called "encrypted_data".

```
openssl pkeyutl -pubin -inkey pubkey.pem -in ./clear_data -encrypt -out ./encrypted_data -pkeyopt
rsa_padding_mode:oaep
```

Decrypt the "encrypted_data" file using the HSM stored private key and output the results to a file called "clear_data2".

```
openssl pkeyutl -engine pkcs11 -keyform engine -inkey "pkcs11:object=my_rsa2048_key" -decrypt -in
./encrypted_data -out ./clear_data2 -pkeyopt rsa_padding_mode:oaep
```

Confirm that the contents of the "clear_data" and "clear_data2" files are identical.

```
diff clear_data clear_data2
```

Example 3: Sign a data file using the HSM stored private key and verify the signature using the public key

Sign the "clear_data" file using the HSM stored private key and output the signature to a file called "clear_data.sig".

```
openssl pkeyutl -engine pkcs11 -keyform engine -inkey "pkcs11:object=my_rsa2048_key" -sign -in
./clear_data -out ./clear_data.sig
```

Verify the signature using the public key.

```
openssl pkeyutl -pubin -inkey pubkey.pem -verify -in ./clear_data -sigfile ./clear_data.sig
```

A message should be output to the screen confirming that the signature was verified successfully.

Example 4: Create a Self-Signed Root Certificate Authority (CA)

Generate a self-signed CA certificate with the HSM stored private key.

```
openssl req -new -x509 -engine pkcs11 -keyform engine -key "pkcs11:object=my_rsa2048_key" -out ssl-
ca-cert.pem -days 365
```

It will prompt for information about the self-signed CA certificate. Once all fields have been entered, it will output to a file called "ssl-ca-cert.pem".

Example 5: Generate a Certificate Signing Request (CSR)

Generate a CSR with the HSM stored private key.

```
openssl req -new -engine pkcs11 -keyform engine -key "pkcs11:object=my_rsa2048_key" -out ssl-cli-
ent-cert-req.pem -days 365
```

It will prompt for information about the certificate. Once all fields have been entered, the certificate signing request will be output to a file called "ssl-client-cert-req.pem".

Example 6: Sign a CSR using the HSM stored private key

Sign a CSR using the HSM stored private key.

```
openssl x509 -req -engine pkcs11 -in ssl-client-cert-req.pem -CA ssl-ca-cert.pem -CAkeyform engine
-CAkey "pkcs11:object=my_rsa2048_key" -CAcreateserial -out signed-client-cert.pem -days 365
```

The signed certificate will be output to a file called "signed-client-cert.pem".

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