MICROSOFT SIGNTOOL

Integration Guide

Applicable Devices:

Vectera Plus

TABLE OF CONTENTS

[1] DOCUMENT INFORMATION

[1.1] DOCUMENT OVERVIEW

The purpose of this document is to provide information regarding the configuration of the Futurex Vectera Plus HSM with Microsoft SignTool using the Futurex CNG library. For additional questions related to your HSM, see the relevant administrator’s guide.

[1.2] APPLICATION DESCRIPTION

From Microsoft's developer documentation website: "SignTool is a command-line tool that digitally signs files, verifies the signatures in files, and timestamps files."

[1.3] PURPOSE OF THE INTEGRATION

Microsoft SignTool uses a code signing certificate to digitally sign files, verify signatures in files, and timestamp files. By integrating with the Vectera Plus, the private key of the code signing certificate is stored on the Vectera Plus and used when SignTool signing operations are performed.

[2] PREREQUISITES

Supported Hardware:

- Vectera Plus, 6.7.x.x and above

Supported Operating Systems:

- Windows 7 and above

Other:

- OpenSSL
- Microsoft SignTool (**NOTE:** SignTool is available as part of the Windows SDK, which you can download from https://developer.microsoft.com/windows/downloads/windows-10-sdk/)

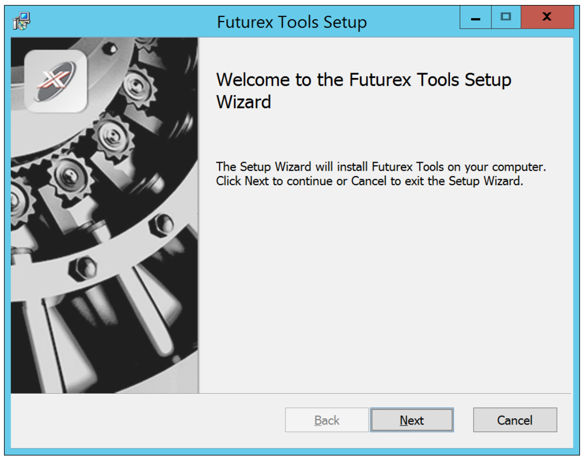
[3] INSTALL FUTUREX CNG (FXCNG)

In a Windows environment, the easiest way to install the Futurex CNG (FXCNG) module is through installing **FXTools**. FXTools can be downloaded from the Futurex Portal. Step by step installation instructions are provided below:

NOTE: The Futurex CNG module needs to be installed on the computer/server where Microsoft SignTool will be used.

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

- 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.

- **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.

[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.

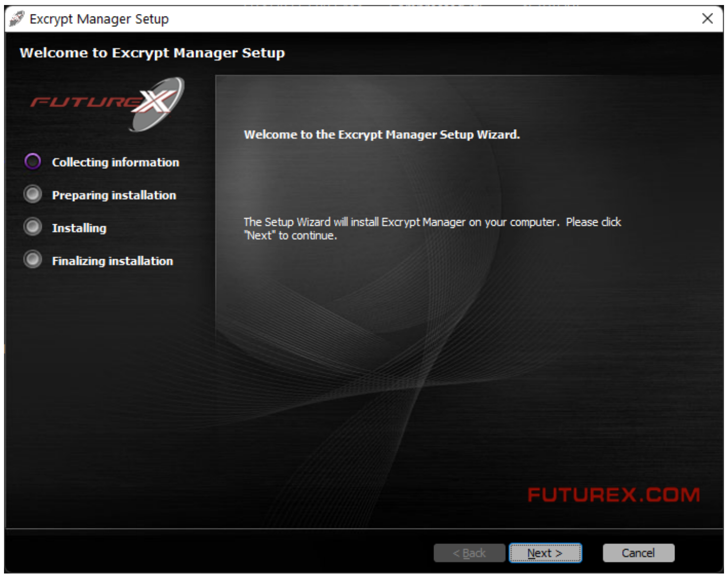
Note: If you will be configuring the Vectera Plus from a Linux computer, you can skip this section. If you will be configuring the Vectera Plus from a Windows computer, installing FXCLI in the next section is still required because FXCLI is the only method that can be used to configure TLS certificates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

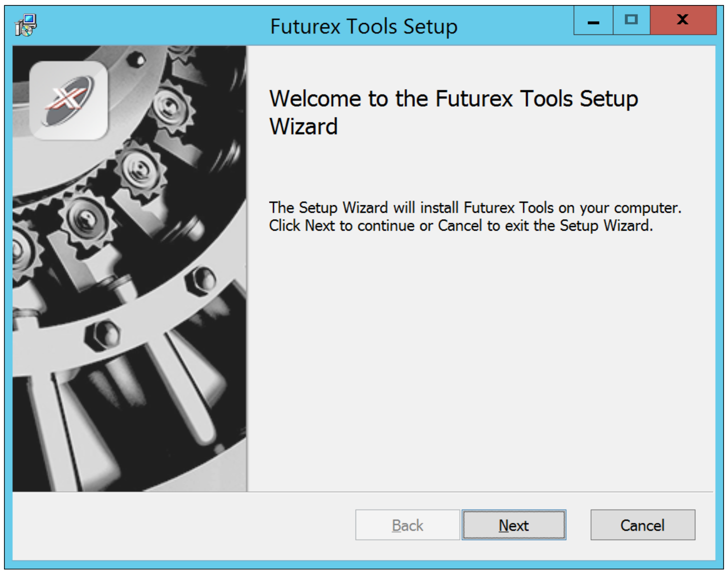
[5] INSTALL FUTUREX COMMAND LINE INTERFACE (FXCLI)

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. 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

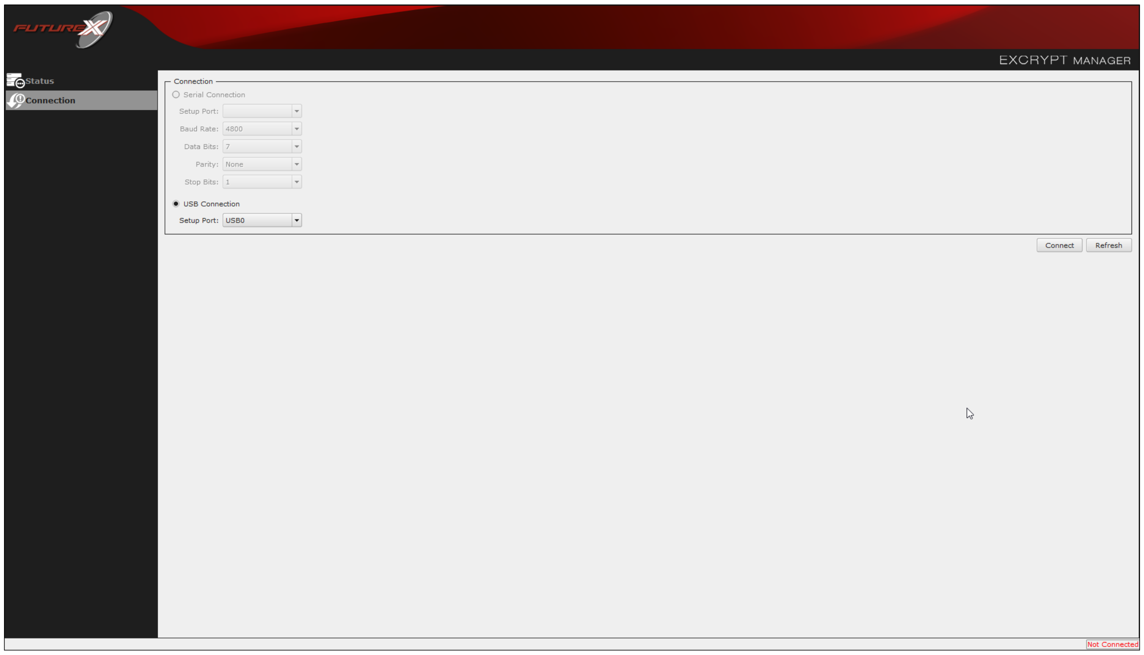
Each of these action items is detailed in the following subsections.

[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

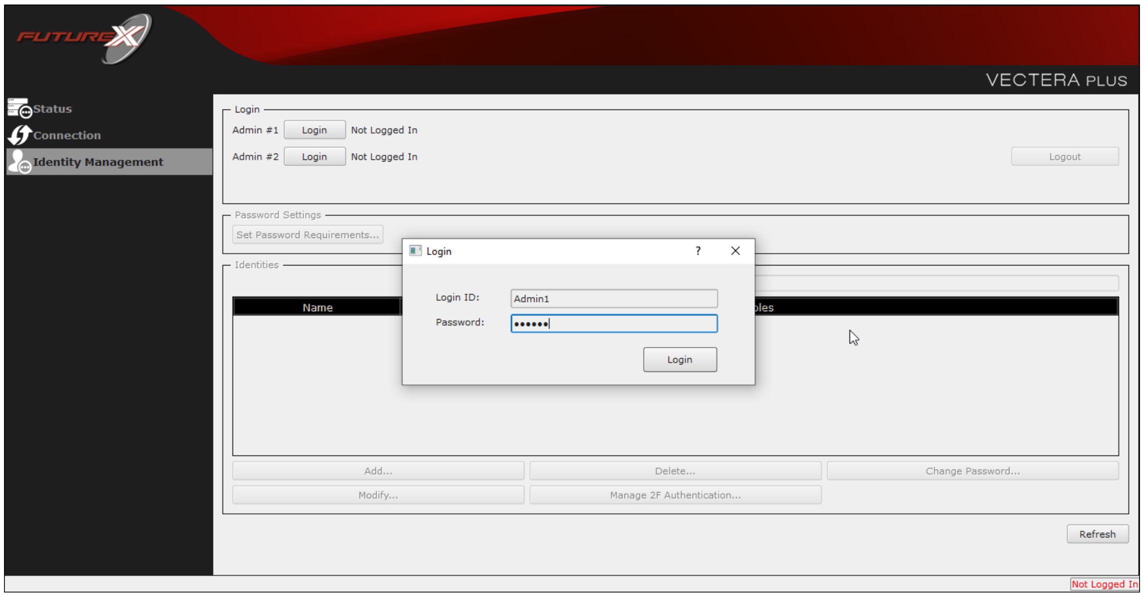
Open Excrypt Manager, click "Refresh" in the lower right-hand side of the Connection menu. Then select "USB Connection" and click "Connect".



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.



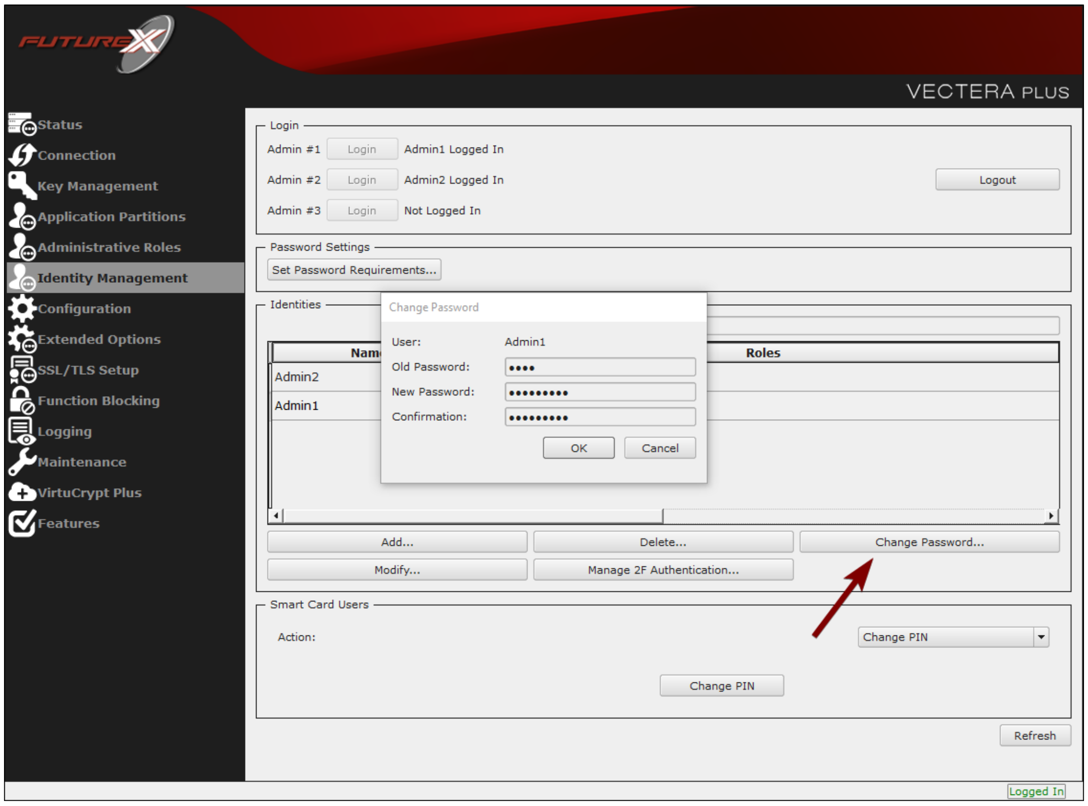
Login
- Admin #1 Login Not Logged In
- Admin #2 Login Not Logged In

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

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").



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)

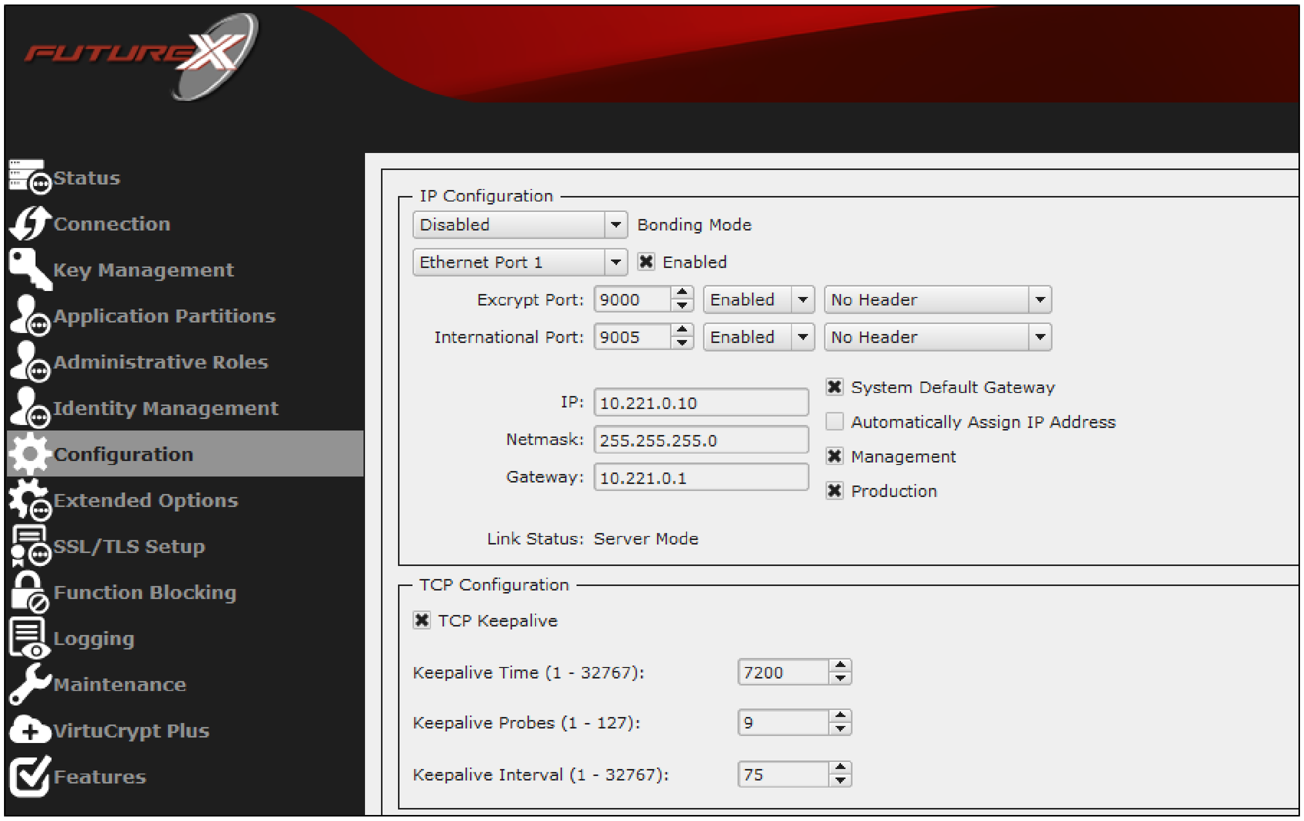
NOTE: For additional information about how to update features on your HSM, please refer to your HSM Administrator's Guide, section **"Download Feature Request File"**.

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.

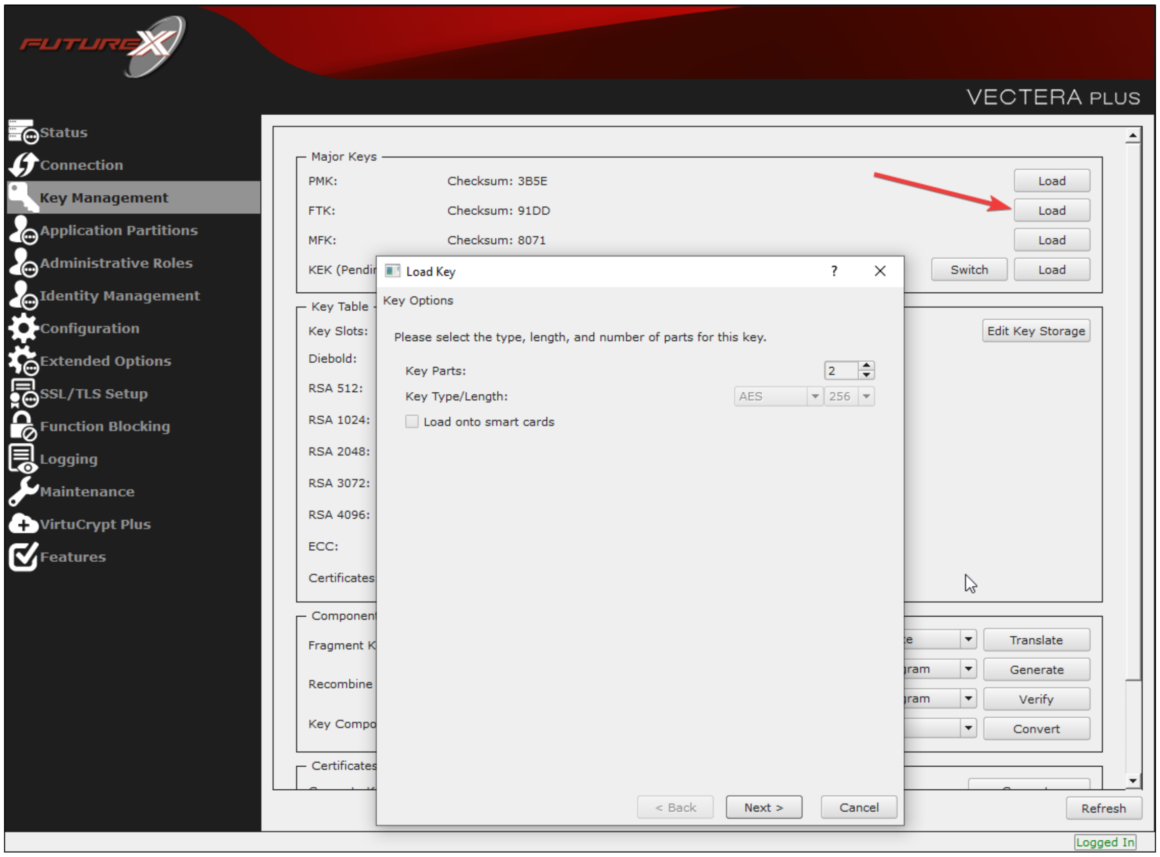
[6.4] LOAD FUTUREX KEY (FTK)

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
KEK (Pendir

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
```

[6.5] 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.

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:

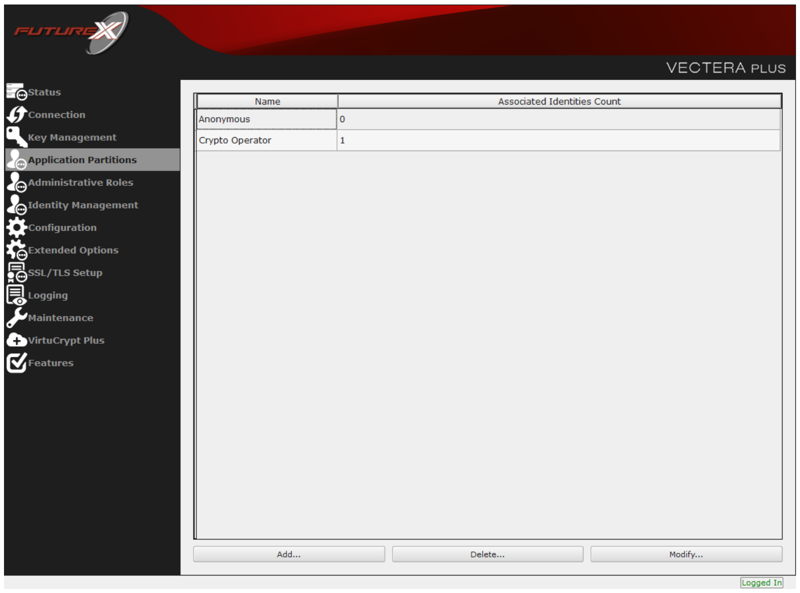
```
$ role modify --name Anonymous --clear-perms --clear-key-ranges
```

```
$ 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

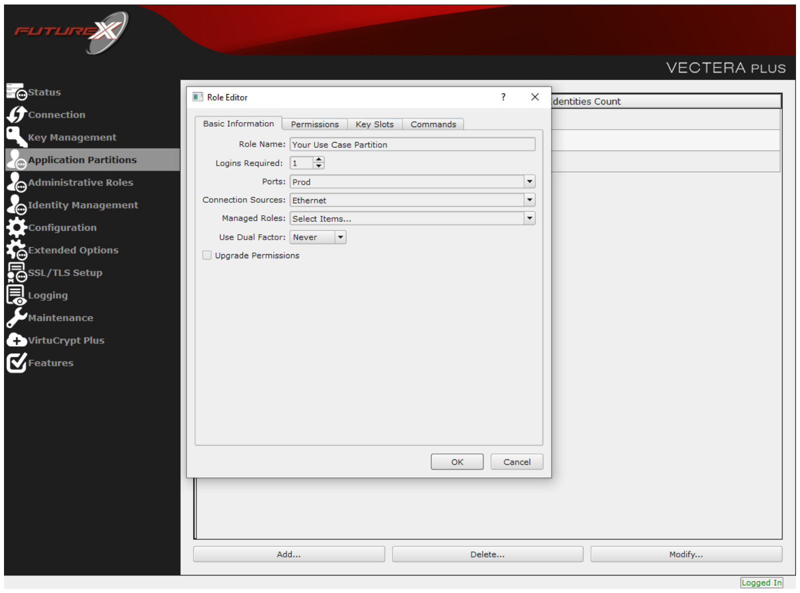
In order for application segregation to occur on the HSM, an Application Partition must be created specifically for your use case. Application partitions are used to segment the permissions and keys on an HSM between applications. The process for configuring a new application partition is outlined in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

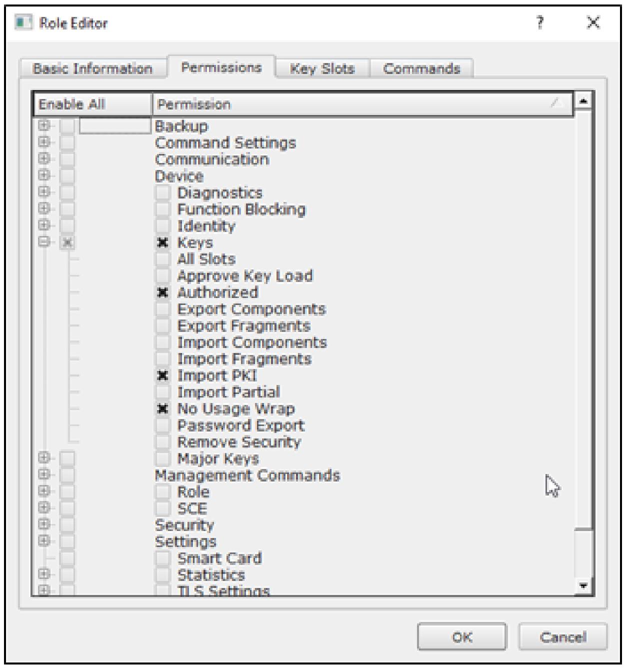
Fill in all of the fields in the *Basic Information* tab exactly how you see below (except for the *Role Name* field). In the *Role Name* field, specify any name that you would like for this new Application Partition. *Logins Required* should be set to "1". *Ports* should be set to "Prod". *Connection 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

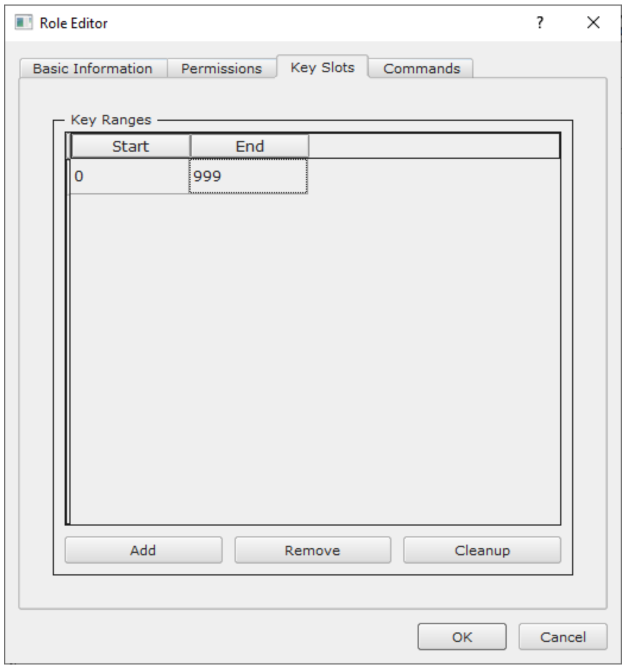
Upgrade Permissions

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
✓ Keys
All Slots
Approve Key Load
✓ Authorized
Export Components
Export Fragments
Import Components
Import Fragments
✓ Import PKI
Import Partial
✓ 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. For the Microsoft Signtool integration, the commands listed below need to be enabled. These can be enabled under the "Commands" tab.

PKCS #11 Communication Commands

- **ECHO**: Communication Test/Retrieve Version
- **HASH**: Retrieve device serial
- **GPKM**: Retrieve key table information

Key Operations Commands

- **RPFP**: Get public components from RSA private key
- **APFP**: Get public components from ECC private key

Signing Commands

- **RSAS**: Generate a signature using RSA private key
- **ASYS**: Generate a signature using PKI 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:

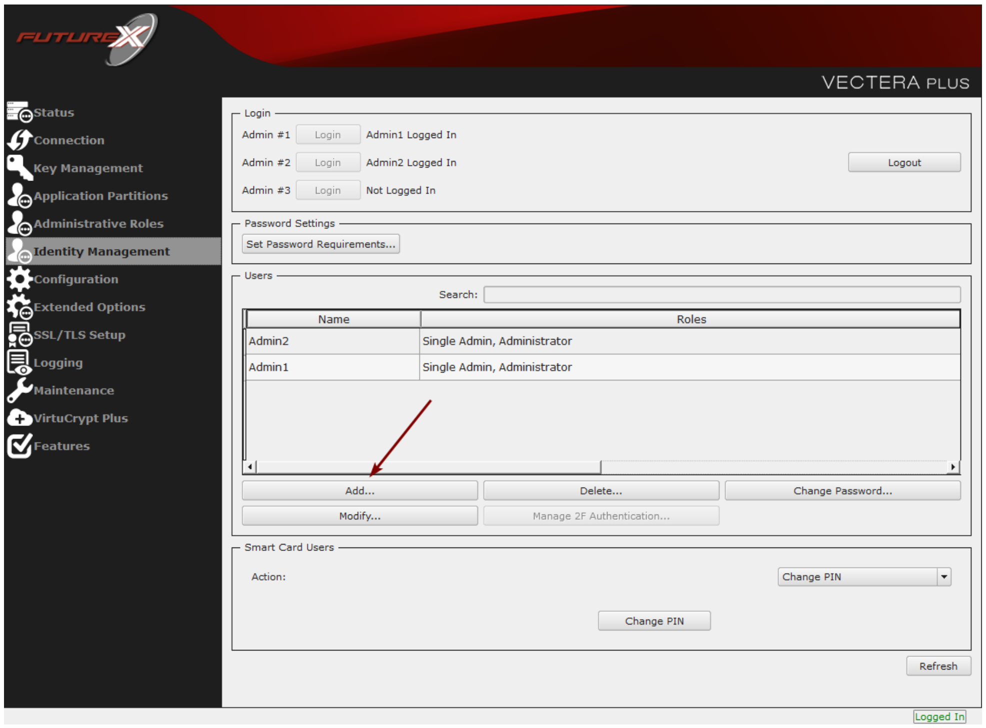
```
$ role add --name Role_Name --application --key-range (0,999) --perm "Keys:Authorized" --perm "Key-
s:Import PKI" --perm "Keys:No Usage Wrap"
```

```
$ role modify --name [role_name] --clear-perms --add-perm Excrypt:ECHO --add-perm Excrypt:HASH --
add-perm Excrypt:GPKM --add-perm Excrypt:RPFP --add-perm Excrypt:APFP --add-perm Excrypt:RSAS --
add-perm Excrypt:ASYS
```

[6.6] 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.*

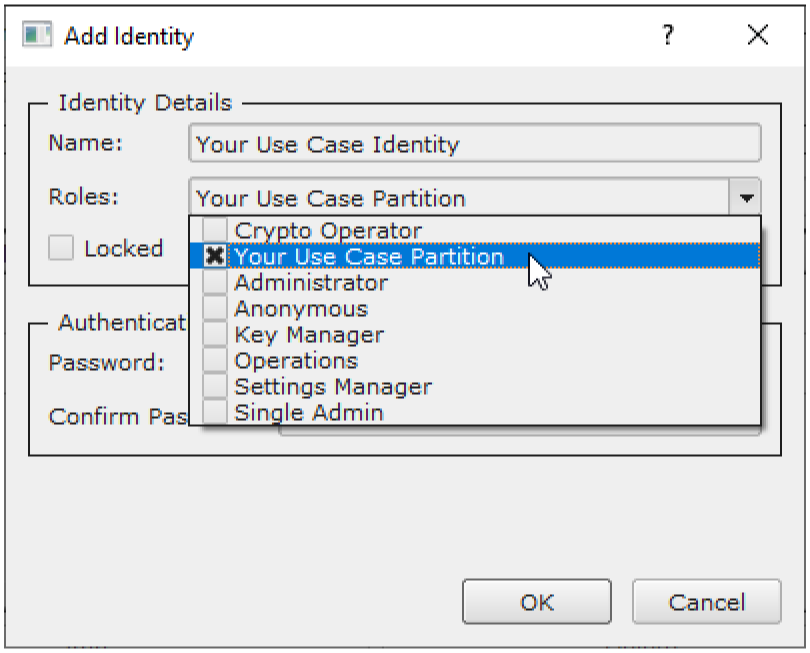
A new identity must be created, which will need to be associated with the Application Partition created in the previous step. To create this new identity, go to *Identity Management*, and click "Add".



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.



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>
```

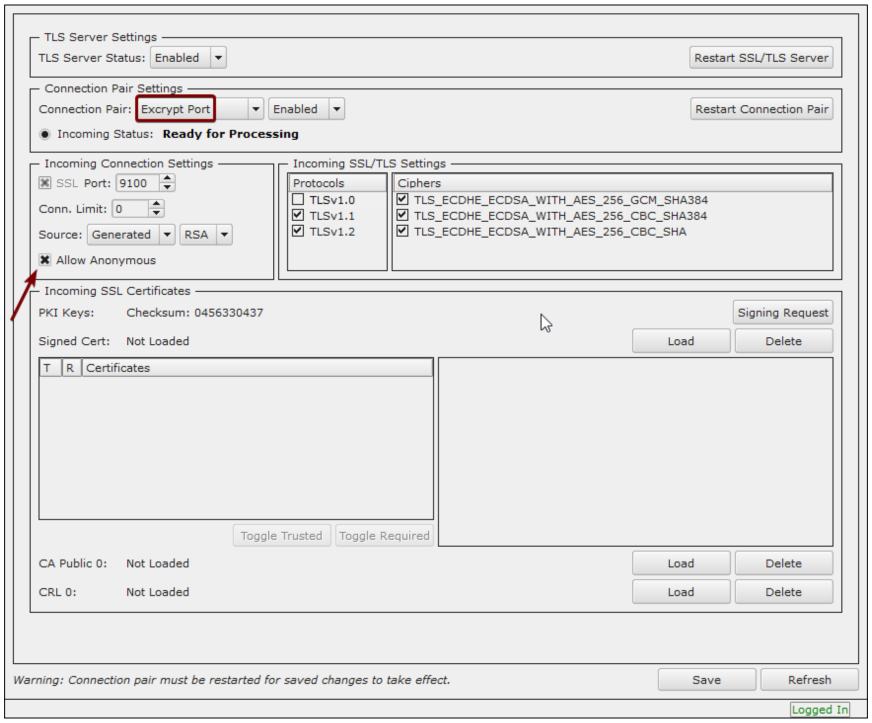
NOTE: Crypto Operator in the fxpkcs11.cfg file must match exactly the name of the identity created in the HSM.

[6.7] 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 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 FUTUREX CNG (FXCNG) CONFIGURATION FILE

The *fxcng.cfg* file allows the user to define specific configurations and set connection information for communication between the FXCNG library and the Vectera Plus. To edit, run a text editor as an Administrator and edit the configuration file accordingly.

NOTE: Our CNG library expects the CNG config file to be in a certain location (*C:\Program Files\Futurex\fxcng\fxcng.cfg*).

[7.1] DEFINE SPECIFIC CONFIGURATIONS

For the Microsoft Signtool integration, the following defines must be set in the **<CONFIG>** section of the *fxcng.cfg* file, as shown below:

```
<UNIQUE-CONNECTIONS>         NO                  </UNIQUE-CONNECTIONS>

<LOGOUT-ON-SESSION-CLOSE>    NO                  </LOGOUT-ON-SESSION-CLOSE>
```

[7.2] DEFINE CONNECTION INFORMATION

In the **<HSM>** section of the FXCNG configuration file, connection information must be defined.

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

    # Login username
    <CRYPTO-OPR>            crypto1                 </CRYPTO-OPR>
    # Automatically login on session open
       <CRYPTO-OPR-PASS>       safest                  </CRYPTO-OPR-PASS>

    # Connection information
    <ADDRESS>               10.0.8.20               </ADDRESS>
    <PROD-PORT>             9100                    </PROD-PORT>
    <PROD-TLS-ENABLED>      YES                     </PROD-TLS-ENABLED>
    <PROD-TLS-ANONYMOUS>    NO                      </PROD-TLS-ANONYMOUS>
    <PROD-TLS-CA>           /home/bbarrows/tls/root.pem        </PROD-TLS-CA>
    <PROD-TLS-CERT>         /home/bbarrows/tls/signed-client-cert.pem      </PROD-TLS-CERT>
    <PROD-TLS-KEY>          /home/bbarrows/tls/PKI.p12   </PROD-TLS-KEY>
    <PROD-TLS-KEY-PASS>     safest                 </PROD-TLS-KEY-PASS>

    # YES = This is communicating through a Guardian
    <FX-LOAD-BALANCE>       NO                      </FX-LOAD-BALANCE>
</HSM>
```

The **<SLOT>** field can be left as the default value of "0".

The **<LABEL>** field can be left as the default value of "Futurex".

In the **<CRYPTO-OPR>** field, specify the name of the identity that was created on the Vectera Plus.

In the **<CRYPTO-OPR-PASS>** field, specify the password of the identity configured in the **<CRYPTO-OPR>** field.

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

In the **<PROD-PORT>** field, set the CNG library to connect to the default TLS production port on the Vectera Plus, port 9100.

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

The **<PROD-TLS-ANONYMOUS>** field defines whether the CNG library should authenticate to the HSM or not. If server-side authentication was configured, this value should be set to "YES". If mutual authentication was configured, this value should be set to "NO".

The location of the CA certificate/s needs to be defined with one or more instances of the **<PROD-TLS-CA>** tag.

The location of the signed client certificate needs to be defined with the **<PROD-TLS-CERT>** tag.

The **<PROD-TLS-KEY>** tag 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 Vectera Plus devices 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”.

For additional details, reference the Futurex CNG technical reference found on the Futurex Portal.

[8] VERIFY THAT THE FUTUREX CNG PROVIDER IS PROPERLY CONFIGURED

1. In a command prompt, execute:

```
certutil -csptest -csp "Futurex CNG" RSA
```

2. If you see the following text, the module is installed properly:

```
Provider Name: Futurex CNG
      Name:  Provider Module:
      UM(1): fxcng.dll
      0(1): 10001, 1
        0: KEY_STORAGE
...
      Name:  Signature Algorithms:
   RSA
    BCRYPT_ASYMMETRIC_ENCRYPTION_INTERFACE -- 3
    NCRYPT_ASYMMETRIC_ENCRYPTION_OPERATION -- 4
    NCRYPT_SIGNATURE_OPERATION -- 10 (16)

    NCryptCreatePersistedKey(Futurex CNG, RSA)
...
  All Algorithms:
    RSA

CertUtil: -csptest command completed successfully.
```

If you do not see the above text, the module is not installed or configured correctly. Review the logs for additional information. The location of the log file is defined in the configuration file in the previous step.

[9] CREATE A CODE SIGNING CERTIFICATE

This section describes two different methods for issuing/importing a code signing certificate on the Vectera Plus:

1. Issued by a CA on the Vectera Plus
2. Importing an existing code signing certificate as a PKCS #12 file onto the Vectera Plus

Microsoft Signtool will subsequently be able to use the code signing certificate to sign files using the private key that is stored on the Vectera Plus.

[9.1] ISSUED BY A CA ON THE VECTERA PLUS

In the following subsection, **Futurex Command Line Interface (FXCLI)** will be used to create a new Certificate Authority (CA) on the Vectera Plus, which will be used to issue a code signing certificate.

1. Run the **fxcli-hsm** program.
2. Connect to the Admin TLS port on the HSM using the **connect tcp** command.

   NOTE: Before connecting, TLS certificates need to be configured in FXCLI using the **tls** commands. For additional details, reference the FXCLI HSM technical reference found on the Futurex Portal.
3. Log in with both default Admin identities. This command will prompt for the username and password. It needs to be run twice (i.e., once for Admin1 and once for Admin2).

   ```
   $ login user
   ```
4. Run the following command to generate a new key in the next available key slot on the Vectera Plus. This key will be used in the next step to create a self-signed CA.

   ```
   $ generate --algo RSA --bits 2048 --usage mak --name SigntoolCaKeyPair --slot next
   ```
5. Run the following command to create a CA certificate using the key that was generated on the HSM in the previous step:

   ```
   $ x509 sign --private-slot SigntoolCaKeyPair --key-usage DigitalSignature --key-usage KeyCer-
   tSign --ca true --dn 'O=Futurex\CN=Signtool CA' --out C:\Integration-Testing-Sand-
   box\SigntoolCa.pem
   ```

   NOTE: The command above will output the CA certificate to the location specified in the `--out` flag.
6. Run the following command to generate a new key in the next available key slot on the Vectera Plus. This key will be used to create a **Certificate Signing Request (CSR)** for the code signing certificate.

   ```
   $ generate --algo RSA --bits 2048 --usage mak --name CodeSigningKeyPair --slot next
   ```
7. Run the following command to assign the value, "CodeSigningKeyPair", to the "label" PKCS #11 attribute of the key created in the previous step:

   ```
   $ keytable extdata --slot 1 --p11-attr label --p11-value "CodeSigningKeyPair"
   ```

NOTE: The value set in the `--slot` flag needs to match the key slot where the "CodeSigningKeyPair" was created.

8. Run the following command to generate a CSR using the "CodeSigningKeyPair":

```
$ x509 req --private-slot CodeSigningKeyPair --out C:\Integration-Testing-Sand-
box\CodeSigning.csr
```

9. Run the following command to issue a code signing certificate using the CA certificate created in step 5:

```
$ x509 sign --private-slot SigntoolCaKeyPair --issuer C:\Integration-Testing-Sand-
box\SigntoolCa.pem --key-usage DigitalSignature --key-usage NonRepudiation --eku CodeSigning -
-ca false --dn 'O=Futurex\CN=Code Signing' --csr C:\Integration-Testing-Sand-
box\CodeSigning.csr --out C:\Integration-Testing-Sandbox\CodeSigning.pem
```

NOTE: The code signing certificate and CA certificate need to be moved to the computer where Microsoft Signtool will be utilized.

[9.2] IMPORTING AN EXISTING CODE SIGNING CERTIFICATE AS A PKCS #12 FILE

In the following subsection, **Futurex Command Line Interface (FXCLI)** will be used to import an existing code signing certificate that is in PKCS #12 format onto the Vectera Plus.

NOTE: The code signing certificate PKCS #12 file must only contain the code signing certificate and it's associated private key. CA certificates should not be included within the file.

1. Run the **fxcli-hsm** program.
2. Connect to the Admin TLS port on the HSM using the **connect tcp** command.

   NOTE: Before connecting, TLS certificates need to be configured in FXCLI using the **tls** commands. For additional details, reference the FXCLI HSM technical reference found on the Futurex Portal.

3. Log in with both default Admin identities. This command will prompt for the username and password. It needs to be run twice (i.e., once for Admin1 and once for Admin2).

```
$ login user
```

4. Run the following command to import the private key of an existing code signing certificate that is in PKCS #12 format (**NOTE:** The code signing certificate that is passed into this command must be in PKCS #12 format, as this format will contain the private key of the code signing certificate within the file, encrypted under a password.):

```
$ pkcs12 import --file C:\Integration-Testing-Sandbox\code_signing_cert.p12 --password safest
--slot next --label ImportedCodeSigningKeyPair --win-system-dacl
```

NOTE: PEM formatted versions of the code signing certificate and CA certificate need to be acquired and then moved to the computer where Microsoft Signtool will be utilized.

[10] IMPORTING CERTIFICATES INTO WINDOWS CERTIFICATE STORE

This section will explain how to import the code signing certificate (which Microsoft SignTool will use to sign files) and the CA certificate(s) that issued the code signing certificate into the Windows certificate store.

[10.1] IMPORT THE CA CERTIFICATE(S)

1. In Windows, open the "Manage computer certificates" application.
2. Right-click on the "Trusted Root Certificate Authorities" certificate store, then select *All Tasks -> Import*.
3. The Local Machine store should be selected. Click **Next**.
4. Click the **Browse** button, then find and select the CA certificate(s) that signed the code signing certificate that SignTool will be using. Click **Next**.
5. Select the option that will place all certificates in the "Trusted Root Certificate Authorities" store, then click **Next**.
6. Click **Finish**.

[10.2] IMPORT THE CODE SIGNING CERTIFICATE

1. In Windows, open the "Manage computer certificates" application.
2. Right-click on the "Personal" certificate store, then select *All Tasks -> Import*.
3. The Local Machine store should be selected. Click **Next**.
4. Click the **Browse** button, then find and select the code signing certificate that SignTool will be using. Click **Next**.
5. Select the option that will place all certificates in the "Personal" store, then click **Next**.
6. Click **Finish**.

[11] ASSOCIATING A PRIVATE KEY WITH A CERTIFICATE

This section will explain how to associate a private key (stored on the Vectera Plus) with it's corresponding code signing certificate (stored in the Local Machine Windows certificate store).

The primary method of associating a private key with a certificate is via a tool called CertUtil. The primary resource for using the CertUtil command past any advanced case is this manual page. However, generally your use-case won't go past the command demonstrated below.

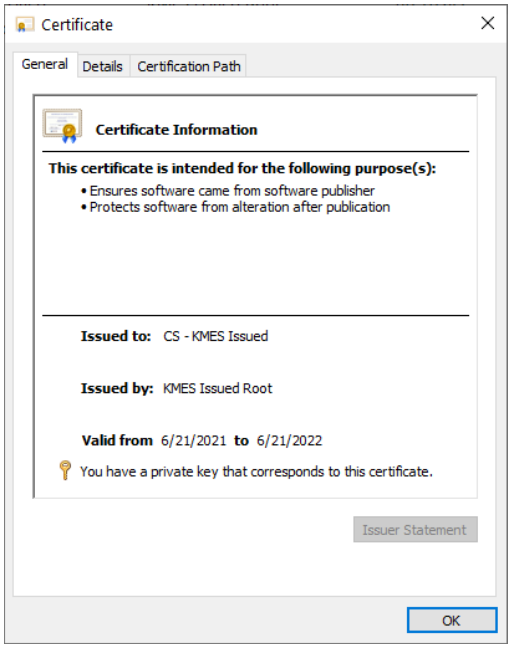
To associate a private key held in the Vectera Plus with a code signing certificate held in the Local Machine Windows certificate store, open the Command Prompt application as an Administrator and run the following command (replacing the fields surrounded in < and > symbols with the actual values that are required):

```
certutil -f -csp <human name of csp> -repairstore <store name> <serial number of cert in store>
```

As an example, the command could look like this:

```
certutil -f -csp "Futurex CNG" -repairstore My 00f204e900000070
```

For this integration, the CSP should be "Futurex CNG" and the store name should be "My". The "My" value tells the CertUtil command to look for the certificate in the X.509 certificate store for personal certificates, which is where we imported the code signing certificate in the previous section. The only field that should be changed is the serial number field. To find the serial number of your certificate, locate it in the Personal certificate store and double click on it. This will pull up the following dialog:



Certificate Information

This certificate is intended for the following purpose(s):
- Ensures software came from software publisher
- Protects software from alteration after publication

Issued to: CS - KMES Issued

Issued by: KMES Issued Root

Valid from 6/21/2021 to 6/21/2022

You have a private key that corresponds to this certificate.

Select the *Details* tab, then note down the serial number that is listed for the certificate to use in the CertUtil command.

[12] TESTING MICROSOFT SIGNTOOL COMMANDS

In this section we'll run two Microsoft SignTool commands (i.e. `signtool sign` and `signtool verify`).

NOTE: The `signtool sign` command pertains more specifically to this integration, as it is the only SignTool command that initiates communication with the Vectera Plus. SignTool must be able to access the private key that is stored on the Vectera Plus in order to complete the code signing operation successfully.

[12.1] SIGN A FILE USING THE CONFIGURED CODE SIGNING CERTIFICATE

NOTE: In the following example an `.exe` file is being signed, but several other types of files can be signed using SignTool. Please reference the following url for details: https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography-tools

Open the Windows Command Prompt application and run the following command:

NOTE: Replace "MyCertificate" with the Subject Name of your certificate and "example.exe" with the name of the file that you are signing.

```
signtool sign /sm /fd sha256 /s My /n "MyCertificate" example.exe
```

If the command is successful you should receive the following message:

```
Done Adding Additional Store
Successfully signed: example.exe
```

[12.2] VERIFY THE FILE THAT WAS SIGNED

To verify the file that was signed run the following command:

```
signtool verify /pa example.exe
```

If the command is successful, you should see output similar to the following:

```
File: example.exe
Index  Algorithm  Timestamp
========================================
0      sha1       None

Successfully verified: example.exe
```

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