



APACHE HTTP SERVER

Integration Guide

Applicable Devices:

Vectera Plus



THIS DOCUMENT CONTAINS CONFIDENTIAL INFORMATION PROPRIETARY TO FUTUREX, LP. ANY UNAUTHORIZED USE, DISCLOSURE, OR DUPLICATION OF THIS DOCUMENT OR ANY OF ITS CONTENTS IS EXPRESSLY PROHIBITED.

TABLE OF CONTENTS

[1] DOCUMENT INFORMATION	3
[1.1] DOCUMENT OVERVIEW	3
[1.2] APPLICATION DESCRIPTION	3
[1.3] GUARDIAN INTEGRATION	3
[2] PREREQUISITES	5
[3] INSTALL FUTUREX PKCS #11 (FXPKCS11)	6
[3.1] INSTALLING THE FXPKCS11 MODULE USING FXTOOLS IN WINDOWS	6
[3.2] INSTALLING THE FXPKCS11 MODULE IN LINUX	7
[4] INSTALL EXCRYPT MANAGER (IF USING WINDOWS)	8
[5] INSTALL FUTUREX COMMAND LINE INTERFACE (FXCLI)	9
[5.1] INSTALLING FXCLI IN WINDOWS	9
[5.2] INSTALLING FXCLI IN LINUX	9
[6] CONFIGURE THE VECTERA PLUS	11
[6.1] CONNECT TO THE HSM THROUGH THE FRONT USB PORT	11
[6.2] REQUIRED FEATURES IN HSM	12
[6.3] NETWORK CONFIGURATION (SETTING THE HSM IP ADDRESS)	12
[6.4] LOAD FUTUREX KEY (FTK)	13
[6.5] CONFIGURE A TRANSACTION PROCESSING CONNECTION AND CREATE AN APPLICATION PARTITION	13
[6.6] CREATE A NEW IDENTITY AND ASSOCIATE IT WITH THE NEWLY CREATED APPLICATION PARTITION	16
[6.7] CONFIGURE TLS AUTHENTICATION	17
[6.8] ENABLE THE SVX MULTI-USAGE COMBINATION FOR ASYMMETRIC KEYS	19
[7] EDIT THE FUTUREX PKCS #11 CONFIGURATION FILE	20
[8] OPENSLL ENGINE INSTALLATION AND CONFIGURATION	22
[8.1] INSTALL LIBP11 AND OPENSLL	22
[8.2] EDIT THE OPENSLL CONFIGURATION FILE	22
[9] APACHE HTTP SERVER CONFIGURATION	24
[9.1] SET FUTUREX PKCS #11 ENVIRONMENT VARIABLES	24
[9.2] CREATE A KEY PAIR ON THE VECTERA PLUS USING PKCS11-TOOL	24
[9.3] GENERATE A CERTIFICATE SIGNING REQUEST (CSR) USING THE APACHE SERVER PRIVATE KEY	25
[9.4] CREATE A SELF-SIGNED ROOT CERTIFICATE AUTHORITY (CA)	25
[9.5] SIGN THE APACHE SERVER CSR	25
[9.6] CONFIGURE APACHE TO USE THE SIGNED CERTIFICATE AND PRIVATE KEY STORED ON THE HSM	25
[9.7] CREATE A CLIENT CERTIFICATE FOR THE BROWSER THAT CONNECTS TO APACHE HTTP SERVER	26
[9.8] CONFIRM THAT APACHE USES THE NEW TLS CERTIFICATE AND PRIVATE KEY (STORED ON THE HSM) FOR HTTPS CONNECTIONS	27
APPENDIX A: XCEPTIONAL SUPPORT	28

[1] DOCUMENT INFORMATION

[1.1] DOCUMENT OVERVIEW

This document provides information about configuring Futurex HSMs with Apache HTTP Server using Futurex PKCS #11 libraries. For additional questions related to your HSM, see the relevant user guide.

[1.2] APPLICATION DESCRIPTION

[1.2.1] About Apache HTTP Server

Apache HTTP Server, typically referred to as Apache, is a free and open-source cross-platform web server software. Originally released in 1995, it is one of the oldest and most reliable web server software on the internet, running 67% of all web servers in the world. Apache is developed and maintained by an open community of developers with the guidance of the Apache Software Foundation.

[1.2.2] What is a web server?

The purpose of a web server is to serve websites on the internet. It accomplishes this via HTTP protocol. The primary job of all web servers is to accept requests from clients and send a response to that request (such as the components of the page that a visitor wants to see).

[1.2.3] Using HSMs to protect Apache Server private keys

The Apache HTTP Server can work with private keys stored on hardware security modules (HSMs), which helps to prevent the keys' disclosure and man-in-the-middle attacks.

For secure communication with the HTTPS protocol, the Apache HTTP server uses the OpenSSL library. OpenSSL does not support PKCS #11 natively. To utilize HSMs, install the `openssl-pkcs11` package on CentOS or the `libengine-pkcs11-openssl` package in Ubuntu. These packages provide access to PKCS #11 modules through the engine interface. You can use a PKCS #11 URI instead of a regular file name to specify a server key, and use a certificate in the configuration file for the appropriate website.

[1.3] GUARDIAN INTEGRATION

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

- Centralization of device management
- Elimination of points of failure
- Distribution of transaction loads

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

Please see the applicable guide in the Futurex Portal, which covers how to use the Guardian Series 3 to configure HSMs for PKCS #11 integrations.

[2] PREREQUISITES

Supported Hardware:

- Vectera Plus, 6.7.x.x and above

Supported Operating Systems:

- Windows 7 and above
- Linux

Other:

- OpenSSL
- Apache HTTP Server (version 2.4.42 or later for proper mod_ssl support)
- mod_ssl (installed and enabled in Apache)

[3] INSTALL FUTUREX PKCS #11 (FXPKCS11)

In a Windows environment, the easiest way to install the **Futurex PKCS #11 (FXPKCS11)** module is with **Futurex Tools (FXTools)**. You can download FXTools from the Futurex Portal. In a Linux environment, you must download a tarball of the FXPKCS11 binaries from the Futurex Portal and then extract the tar file locally where you want the application to be installed on your system. The following sections provide step-by-step installation instructions for both of these scenarios.

Note: Install FXPKCS11 on the same computer as the application integrating with the Vectera Plus HSM.

[3.1] INSTALLING THE FXPKCS11 MODULE USING FXTOOLS IN WINDOWS

Run the Futurex Tools installer as an administrator and follow the prompts in the setup wizard to complete the installation.

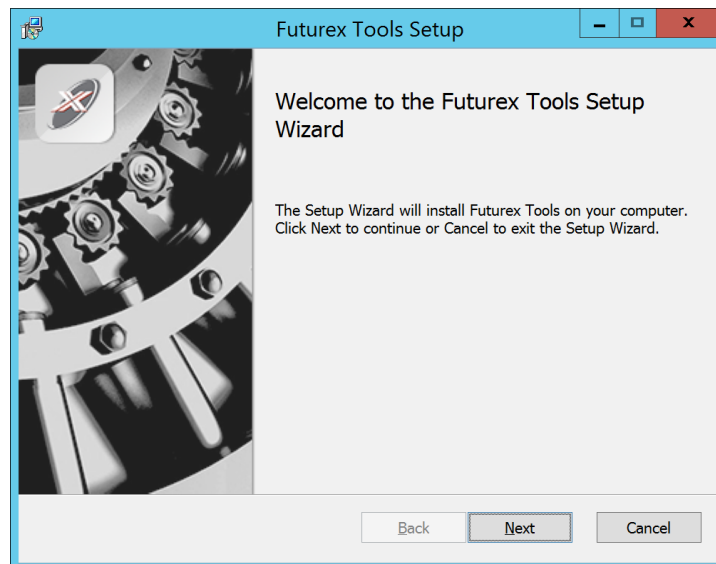


FIGURE: FUTUREX TOOLS SETUP WIZARD

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

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

If the Futurex Secure Access Client was selected, the process will also install the Futurex Excrypt Touch driver, which might start minimized or in the background.

After the installation completes, all services are installed in the C:\Program Files\Futurex\ directory. The CNG Module, CSP Module, EKM Module, and PKCS #11 Module all require configuration files, which are located in their corresponding directory with a .cfg extension. In addition, the installation registers the CNG and CSP Modules in the Windows Registry (HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider), and installs them in the C:\Windows\System32\ directory.

[3.2] INSTALLING THE FXPKCS11 MODULE IN LINUX

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

Note: To make the Futurex PKCS #11 module accessible system-wide, move it to the /usr/local/bin directory as an administrative user. If only the current user needs to use the module, then install it in \$HOME/bin.

The extracted content of the tar file is a single fxpkcs11 directory. Inside the fxpkcs11 directory is the following files and directories:

- **fxpkcs11.cfg:** FXPKCS11 configuration file
- **x86/:** This folder contains the module files for 32-bit architecture
- **x64/:** This folder contains the module files for 64-bit architecture

The x86 and x64 directories each contain two subdirectories, OpenSSL-1.0.x and OpenSSL-1.1.x. These OpenSSL directories contain the following FXPKCS11 module files built with the respective OpenSSL versions:

- **configTest:** Program to test configuration and connection to the HSM
- **libfxpkcs11.so:** FXPKCS11 Library File
- **libfxpkcs11-Debug.so:** FXPKCS11 Debug Library File
- **PKCS11Manager:** Program to test connection and manage the HSM through the FXPKCS11 library

By default, the FXPKCS11 module looks for the FXPKCS11 configuration file (i.e., fxpkcs11.cfg) in the /etc directory. Alternatively, a system environment variable can be defined for the location of the FXPKCS11 configuration file. To do so permanently, open the /etc/profile file in a text editor as an administrative user, add the following line at the bottom, and save the file.

```
export FXPKCS11_CFG=/usr/local/bin/fxpkcs11/fxpkcs11.cfg
```

Note: The file location specified above must be specific to where the FXPKCS11 configuration file is saved on your system.

[4] INSTALL EXCRYPT MANAGER (IF USING WINDOWS)

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

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

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

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

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

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

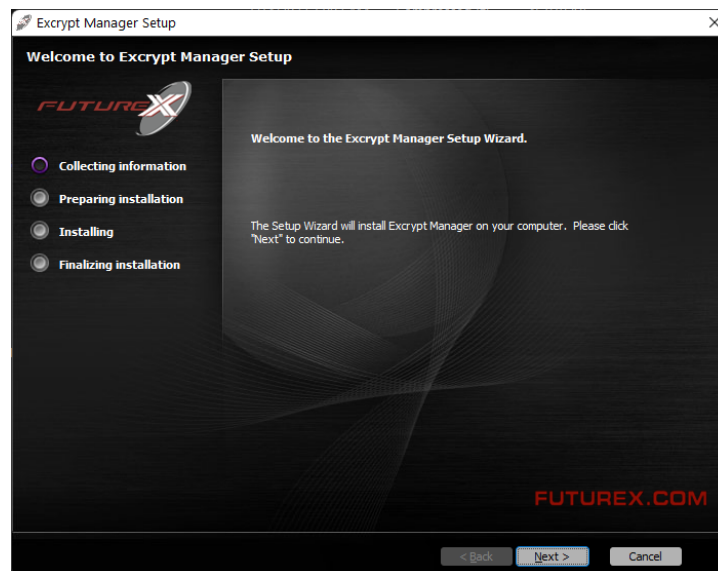


FIGURE: EXCRYPT MANAGER SETUP WIZARD

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.

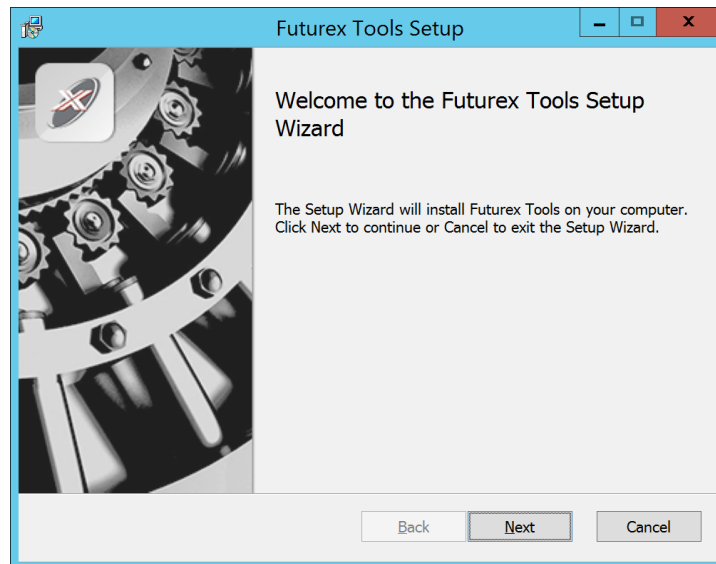


FIGURE: FUTUREX TOOLS SETUP WIZARD

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

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

[5.2] INSTALLING FXCLI IN LINUX

Download FXCLI

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

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

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

Futurex offers the following features for FXCLI:

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

Install FXCLI

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

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

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

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

Running FXCLI

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

```
$ fxcli-hsm
```

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

[6] CONFIGURE THE VECTERA PLUS

To establish a connection between the Futurex PKCS #11 library and the Vectera Plus, perform the following configuration steps:

Note: You can complete all of the steps in this section using either Excrypt Manager or FXCLI (except for section 6.7.2, which can only be completed using FXCLI). Optionally, you can complete steps 4 through 6 using the Guardian Series 3 (Please refer to the applicable guide for configuring HSMs for PKCS #11 integrations using the Guardian Series 3).

1. Connect to the HSM through the front USB port. (**Note:** If you are using a virtual HSM for the integration, you must connect to it over the network through FXCLI, the Excrypt Touch, or the Guardian Series 3):
 - a. Connecting via Excrypt Manager
 - b. Connecting via FXCLI
2. Validate that the correct features are enabled on the HSM.
3. Set up the network configuration.
4. Load the Futurex FTK.
5. Configure a Transaction Processing connection and create a new Application Partition.
6. Create a new identity that has access to the newly created Application Partition.
7. Configure TLS Authentication by using one of the following options:
 - a. Enable server-side authentication.
 - b. Create client certificates for mutual authentication.
8. Enable the SVX multi-usage combination for asymmetric keys

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

[6.1] CONNECT TO THE HSM THROUGH THE FRONT USB PORT

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

Connecting through Excrypt Manager

1. Open Excrypt Manager and click [**Refresh**] in the lower right-hand side of the Connection menu. Then, select **USB Connection** and click [**Connect**].
2. Log in with both default Admin identities.
3. You must change the default Admin passwords for both of your default Admin identities (**Admin1** and **Admin2**) to load the major keys onto the HSM. To do so via Excrypt Manager, open the **Identity Management** menu, select the first default Admin identity (**Admin1**), and select [**Change Password...**]. Enter the old password and enter the new password twice. Select [**OK**]. Perform the same steps for the second default Admin identity (**Admin2**).

Connecting through FXCLI

1. Start the FXCLI application and run the following commands:

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

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

2. You must change the default Admin passwords for both of your default Admin Identities in order to load the major keys onto the HSM. Use the following FXCLI commands to change the passwords for each default Admin Identity.

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

Note: The preceding **user change-password** commands prompt you to enter the old and new passwords.

[6.2] REQUIRED FEATURES IN HSM

To establish a connection between the Futurex PKCS #11 Library and the Vectera Plus, the HSM must be configured with the following features:

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

Note: For additional information about how to update features on your HSM, refer to the “**Download Feature Request File**” section of the Vectera Plus user guide.

Note: Setting the **Command Primary Mode** on the HSM to *General Purpose (GP)* enables the option to create the FTK major key in the HSM. This key is required to be able to use the Futurex PKCS #11 library to communicate with the HSM. For detailed information about how to load major keys on the HSM, refer to the Vectera Plus user guide.

[6.3] NETWORK CONFIGURATION (SETTING THE HSM IP ADDRESS)

Note: For this step you need to be logged in with an identity that has a role with permissions

Communication:Network Settings. You can use the default Administrator role and Admin identities.

Excrypt Manager

1. Navigate to the **Configuration** menu and modify the IP configuration as required.

FXCLI

1. Run the **network interface modify** FXCLI command to set an IP for the HSM. An example is provided below to show the command syntax:

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

Note: At this point during the HSM configuration, consider the following:

- You can complete the remaining HSM configurations in this section using the Guardian Series 3 (see the applicable guide for configuring HSMs for PKCS #11 integrations using the Guardian Series 3), except for the final subsection, which covers creating connection certificates for mutual authentication.
- If you are performing the configuration on the HSM directly right now, but plan to add the HSM to a Guardian later, you might have to synchronize the HSM after you add it to a Device Group on the Guardian.
- If your use-case requires configuration through a CLI, then you should manage the HSMs directly.

[6.4] LOAD FUTUREX KEY (FTK)

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

The FTK wraps all keys stored on the HSM used with PKCS #11. If using multiple HSMs in a cluster, you can use the same FTK for syncing HSMs. An HSM must have an FTK before you can use it with PKCS #11.

Excrypt Manager

1. Navigate to the **Key Management** menu, then select the **Load** button for the FTK in the Major Keys section. You can load keys loaded that are XOR'd together, M-of-N fragments, or generated. If this is the first HSM in a cluster, we recommend you generate the key and save to smart cards as M-of-N fragments.

FXCLI

1. Run the following **majorkey** FXCLI commands to load an FTK into an HSM. You must generate a random FTK if this is the first HSM you are setting up. Optionally, you can also load an FTK onto smart cards simultaneously with the **-m** and **-n** flags, as shown in the following example:

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

If it is a second HSM you're setting up in a cluster, load the FTK from smart cards with the following command:

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

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

Note: For this step you need to be logged in with an identity that has a role with permissions **Role:Add**, **Role:Assign All Permissions**, **Role:Modify**, **Keys:All Slots**, and **Command Settings:Excrypt**. You can use the default Administrator role and Admin identities.

Note: For the purposes of this integration guide, the terms *Application Partition* and *Role* are synonymous.

[6.5.1] Configure a Transaction Processing connection

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

- It should not have access to the **All Slots** permissions.
- It should not have access to any key slots.
- Enable only the PKCS #11 communication commands.

Excrypt Manager

1. Navigate to the **Application Partitions** menu, select the **Transaction Processing** Application Partition, and click [**Modify...**].
2. In the **Permissions** tab, leave the top-level **Keys** permission checked, but uncheck the **All Slots** sub permission.
3. In the **Key Slots** tab, ensure that the settings do not specify key ranges. By default, the Transaction Processing Application Partition has access to the entire range of key slots on the HSM.
4. In the **Commands** tab, make sure to enable only the following PKCS #11 Communication commands:
 - **ECHO**: Communication Test/Retrieve Version
 - **PRMD**: Retrieve HSM restrictions
 - **RAND**: Generate random data
 - **HASH**: Retrieve device serial
 - **GPKM**: Retrieve key table information
 - **GPKS**: General purpose key settings get/change
 - **GPKR**: General purpose key settings get (read-only)

FXCLI

1. Run the following **role modify** FXCLI commands to remove all permissions and key ranges that are currently assigned to the **Transaction Processing** role and enable only the PKCS #11 Communication commands:

Note: The **Transaction Processing** role was previously referred to as the **Anonymous** role. That is why *Anonymous* is specified in the name field in the commands below.

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

```
$ role modify --name Anonymous --add-perm "Keys" --add-perm Excrypt:ECHO --add-perm Excrypt:PRMD --add-perm Excrypt:RAND --add-perm Excrypt:HASH --add-perm Excrypt:GPKM --add-perm Excrypt:GPKS --add-perm Excrypt:GPKR
```

[6.5.2] Create an Application Partition

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

Excrypt Manager

1. Navigate to the **Application Partitions** menu and select [**Add...**].
2. In the **Basic Information** tab, configure all of the fields as follows:
 - a. For **Role Name**, specify any name that you would like for this new Application Partition.
 - b. Set **Logins Required** to *1*.
 - c. Set **Ports** to *Prod*.
 - d. Configure **Connection Sources** to *Ethernet*.
 - e. Leave **Managed Roles** blank because you specify the exact Permissions, Key Slots, and Commands for this Application Partition or Role to have access to.
 - f. Set **Use Dual Factor** to *Never*.
 - g. Leave **Upgrade Permissions** unchecked.
3. In the **Permissions** tab, select the following key permissions:
 - **Keys**
 - **Authorized** (allows for keys that require login)
 - **Import PKI** (allows trusting an external PKI. Generally not recommended, but some applications use this to allow for PKI symmetric key wrapping.)
 - **No Usage Wrap** (allows for interoperable key wrapping without defining key usage as part of the wrapped key. Use this only if you want to exchange keys with external entities or use the HSM to wrap externally used keys.)
4. In the **Key Slots** tab, we recommend you create a range of 1000 total keys that do not overlap with another Application Partition. Within the specified range, you should have ranges for both symmetric and asymmetric keys. If the application requires more keys, configure accordingly.
5. Based on application requirements, particular functions need to be enabled on the Application Partition to use the HSMs functionality. The commands that Apache HTTP Server requires for its operation are listed below. These can be enabled in the **Commands** tab.

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
- **GPKR**: General purpose key settings get (read-only)
- **TIME**: Get/Set the HSM Internal Clock

Key Operations Commands

- **ASYL**: Load asymmetric key into key table
- **GECC**: Generate an ECC Key Pair

Signing Commands

- **ASYS**: Generate a Signature Using a Private Key

FXCLI

1. Run the following **role** FXCLI commands to create the new Application Partition and enable all required functions:

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

```
$ role modify --name [role_name] --clear-perms --add-perm Excrypt:ECHO --add-perm Excrypt:HASH --add-perm Excrypt:GPKM --add-perm Excrypt:GPKS --add-perm Excrypt:GPKR --add-perm Excrypt:TIME --add-perm Excrypt:ASYL --add-perm Excrypt:GECC --add-perm Excrypt:ASYS
```

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

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

Excrypt Manager

1. Navigate to the **Identity Management** menu and select [**Add...**].
2. Specify a name for the new identity and open the **Roles** drop-down menu to select the name of the previously created Application Partition. This associates the new identity with the Application Partition that you created.

FXCLI

1. Run the **identity add** FXCLI command to create a new identity and associate it with the Application Partition/Role that you created:

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

You must set the name of this identity in the fxpkcs11.cfg file, in the following section:

```
#HSM crypto operator identity name
<CRYPTO-OPR>    [insert name of identity that you created]    </CRYPTO-OPR>

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


[6.7] CONFIGURE TLS AUTHENTICATION

Note: For this step you need to be logged in with an identity that has a role with permissions **Keys:All Slots**, **Management Commands:Certificates**, **Management Commands:Keys**, **Security:TLS Sign**, and **TLS Settings:Upload Key**. You can use the default Administrator role and Admin identities.

[6.7.1] Enable server-side authentication (option 1)

Futurex recommends mutually authenticating to the HSM using client certificates, but the Vectera Plus also supports server-side authentication. The following steps outline the process for enabling server-side authentication.

Excrypt Manager

1. Navigate to the **SSL/TLS Setup** menu. Then, select the **Excrypt Port** in the Connection Pair dropdown, check the **Allow Anonymous** box, and click [**Save**].

FXCLI

1. Run the **tls-ports set** FXCLI command to enable server-side authentication with the **Allow Anonymous** SSL/TLS setting:

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

[6.7.2] Create Connection Certificates for mutual authentication (option 2)

As mentioned previously, Futurex recommends mutually authenticating to the HSM using client certificates, and the system enforces mutual authentication by default. In the following example, FXCLI generates a CA which is used to sign the HSM server certificate and a client certificate. The client keys and CSR are generated using OpenSSL.

Note:

- For this example, you must connect the computer that is running FXCLI to the front USB port of the HSM.
 - If you do not specify a file path for commands that create an output file, FXCLI saves the file to the current working directory.
 - Using user-generated certificates requires you to load a PMK on the HSM.
 - If you run **help** by itself, a full list of available commands displays. You can see all of the available options for any given command by running the command name followed by **help**.
1. Enter the FXCLI prompt by running **fxcli-hsm** in a terminal.
 2. Perform the following steps to create connection certificates for mutual authentication:

```
# Connect your laptop to the HSM via the USB port on the front, then run this command.  
$ connect usb
```

```
# Log in with both default Admin identities. This command will prompt for the username and  
password. You will need to run this command twice.  
$ login user
```

```
# Generate a TLS CA and store it in an available key slot on the HSM
$ generate --algo RSA --bits 2048 --usage mak --name TlsCaKeyPair --slot next
```

```
# Create a root certificate
$ x509 sign \
  --private-slot TlsCaKeyPair \
  --key-usage DigitalSignature --key-usage KeyCertSign \
  --ca true --pathlen 0 \
  --dn 'O=Futurex\CN=Root' \
  --out TlsCa.pem
```

```
# Generate the server keys for the HSM
$ tls-ports request --pair "Excrypt Port" --file production.csr --pki-algo RSA
```

```
# Sign the server CSR with the newly created TLS CA
$ x509 sign \
  --private-slot TlsCaKeyPair \
  --issuer TlsCa.pem \
  --csr production.csr \
  --eku Server --key-usage DigitalSignature --key-usage KeyAgreement \
  --ca false \
  --dn 'O=Futurex\CN=Production' \
  --out TlsProduction.pem
```

```
# Push the signed server PKI to the production port on the HSM
$ tls-ports set --pair "Excrypt Port" \
  --enable \
  --pki-source Generated \
  --clear-pki \
  --ca TlsCa.pem \
  --cert TlsProduction.pem \
  --no-anon
```

3. Run the following OpenSSL commands from Windows PowerShell rather than from the FXCLI program to generate client keys and CSR:

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

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

4. Using FXCLI, sign the CSR that was just generated using OpenSSL.

```
# Sign the client CSR under the root certificate that was created
$ x509 sign \
  --private-slot TlsCaKeyPair \
  --issuer TlsCa.pem \
  --csr ClientPki.csr \
  --eku Client --key-usage DigitalSignature --key-usage KeyAgreement \
  --dn 'O=Futurex\CN=Client' \
  --out SignedPki.pem
```

5. Run the remaining commands from Windows PowerShell:

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

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

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

Apache HTTP Server requires asymmetric keys with multiple usages, which can be configured, but is not enabled by default on the Vectera Plus. The specific multi-usage combination that Apache requires is SVX.

Excrypt Manager

1. Navigate to the *Extended Options* menu. In the Usage section, there is a dropdown that allows you to select **Asymmetric Authorize** or **Asymmetric Anonymous**. If you configured mutual authentication for the connection between the Futurex PKCS #11 library and the Vectera Plus in the previous section, select **Asymmetric Authorize**. If you configured server-side authentication in the previous section, select **Asymmetric Anonymous**. After selecting one of the options, click [**Add**].
2. Select the SVX usage combination and click [**Ok**].
3. Click the [**Save**] button on the bottom-right-hand side of the window to save the changes.

FXCLI

1. If you configured mutual authentication for the connection between the Futurex PKCS #11 library and the Vectera Plus in the previous section, run the **multi-usage add** FXCLI command below to add the SVX multi-usage combination for authorized users:

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

If you configured server-side authentication in the previous section, run the **multi-usage add** FXCLI command below to add the SVX multi-usage combination for asymmetric keys for anonymous users:

```
$ multi-usage add --asymmetric --anon -svx
```

[7] EDIT THE FUTUREX PKCS #11 CONFIGURATION FILE

The Futurex PKCS #11 configuration file (i.e., `fxpkcs11.cfg`) is used by the Futurex PKCS #11 library to connect to the HSM. It enables the user to modify certain configurations and set connection details. This section covers the **<HSM>** portion of the `FXPKCS11` config file, where the connection details are set.

Note: By default, the `FXPKCS11` library looks for the configuration file at `C:\Program Files\Futurex\fxpkcs11\fxpkcs11.cfg` for Windows and `/etc/fxpkcs11.cfg` for Linux. Alternatively, the `FXPKCS11_CFG` environment variable can be set to the location of the `fxpkcs11.cfg` file.

Open the `fxpkcs11.cfg` file in a text editor as an administrator and edit it accordingly.

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

# HSM crypto operator user name
<CRYPTO-OPR>    [identity_name] </CRYPTO-OPR>
# Automatically login on session open
#<CRYPTO-OPR-PASS> [identity_password] </CRYPTO-OPR-PASS>

# Connection information
<ADDRESS>      10.0.8.30   </ADDRESS>
<PROD-PORT>    9100       </PROD-PORT>
<PROD-TLS-ENABLED> YES    </PROD-TLS-ENABLED>
<PROD-TLS-ANONYMOUS> NO    </PROD-TLS-ANONYMOUS>
# <PROD-TLS-CA> /home/user/tls/root.pem </PROD-TLS-CA>
# <PROD-TLS-CA> /home/user/tls/sub1.pem </PROD-TLS-CA>
# <PROD-TLS-CA> /home/user/tls/sub2.pem </PROD-TLS-CA>
<PROD-TLS-KEY> /home/user/tls/PKI.p12 </PROD-TLS-KEY>
<PROD-TLS-KEY-PASS> safest </PROD-TLS-KEY-PASS>

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

The **<SLOT>** and **<LABEL>** fields specify PKCS11 slot 0 and the label *Futurex*.

In the **<CRYPTO-OPR>** field, specify the name of the identity you created for the Application Partition.

The **<CRYPTO-OPR-PASS>** field allows you to specify the password of the identity configured in the **<CRYPTO-OPR>** field. This can be used to log the application into the HSM automatically, if required.

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

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

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

The **<PROD-TLS-ANONYMOUS>** field defines whether the `FXPKCS11` library authenticates to the server.

The **<PROD-TLS-KEY>** field defines the location of the client private key. Supported formats for the TLS private key are PKCS #1 clear private keys, PKCS #8 encrypted private keys, or a PKCS #12 file that contains the private key and certificates encrypted under the password specified in the **<PROD-TLS-KEY-PASS>** field.

Because a PKCS #12 file is defined in the **<PROD-TLS-KEY>** field in this example, the signed client cert does not need to be defined with the **<PROD-TLS-CERT>** tag, nor do the CA cert/s need to be defined with one or more instances of the **<PROD-TLS-CA>** tag.

If you use Guardian to manage HSMs in a cluster, define the **<FX-LOAD-BALANCE>** field as *YES*. Otherwise, set it to *NO*.

After you finish editing the `fxpkcs11.cfg` file, run the `PKCS11Manager` file to test the connection against the HSM and check the `fxpkcs11.log` for errors and information. For more information, refer to the Futurex PKCS #11 technical reference found on the Futurex Portal.

[8] OPENSSL ENGINE INSTALLATION AND CONFIGURATION

This section covers the installation and configuration of **libp11**, **OpenSC**, and the **pkcs11 engine plugin** for the OpenSSL library. These libraries provide the following:

- **libp11** provides a higher-level interface to access PKCS #11 objects compared to the PKCS #11 library. 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 you to access PKCS #11 modules in a semi-transparent way.

[8.1] INSTALL LIBP11 AND OPENSC

Ubuntu/Debian

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
```

Red Hat/CentOS

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

```
$ sudo yum check-update
$ sudo yum install openssl-pkcs11
$ sudo yum install opensc
```

[8.2] EDIT THE OPENSSL CONFIGURATION FILE

The following instructions are the same for Ubuntu/Debian-based Linux distributions and Red Hat/CentOS-based Linux distributions, with the exception of the **dynamic_path** define in the **openssl.cnf** file. On Ubuntu 18, the location of the **pkcs11.so** file that needs to be specified in the **dynamic_path** define is **/usr/lib/x86_64-linux-gnu/engines-1.1/pkcs11.so**. On CentOS 8 the location of the **pkcs11.so** file that needs to be specified in the **dynamic_path** define is **/usr/lib64/engines-1.1/pkcs11.so**. To confirm the location of the **pkcs11.so** file on your system, run the following command in a terminal as root:

```
$ find / -name "pkcs11.so"
```

Proceed with the following instructions to edit the OpenSSL configuration file.

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

```
$ openssl version -d
```

Note: If you prefer to edit the global OpenSSL configuration file, its location is most often in `/etc/ssl/openssl.cnf`.

2. In a text editor, open the `openssl.cnf` file that is inside of the OpenSSL directory determined from the previous command.

You must place this line at the top before any sections are defined:

```
openssl_conf = openssl_init
```

Add the following 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 = 0
```

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

Note: You must set the password for the identity created on the Vectera Plus in the PIN field.

[9] APACHE HTTP SERVER CONFIGURATION

[9.1] SET FUTUREX PKCS #11 ENVIRONMENT VARIABLES

1. In a terminal, run the following commands to set the required FXPKCS11 environment variables:

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

Note: Be sure to modify the file path to match the location where the libfxpkcs11.so and fxpkcs11.cfg files are stored on your system.

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

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

```
$ sudo pkcs11-tool --module $FXPKCS11_MODULE --login --keypairgen --key-type EC:prime256v1 --label "apache_ecc_privatekey" --id "123456"
```

Important: At the time of writing, Apache HTTP Server has a bug that prevents RSA certificates from being served correctly to the browser. This bug may have been resolved, but regardless, we recommend that you create and use an ECC certificate as demonstrated.

The above pkcs11-tool command prompts for the user PIN. Enter the password of the identity configured in the fxpkcs11.cfg file. If successful, the command output will list the keys that pkcs11-tool created on the Vectera Plus, as shown below:

```
Key pair generated:
Private Key Object; EC
  label:      apache_ecc_privatekey
  ID:        123456
  Usage:     sign, derive
Public Key Object; EC  EC_POINT 256 bits
  EC_POINT:  04410446d2cd-
c802be18a8b-
cf5e7-
dec6eaalda93d59ca70f12e-
b395ce8b1592f394c02d0bc5adbef3128724ab4b17fded19dfa1148ca7affe443fafdf1b0ad292ba90
  EC_PARAMS: 06082a8648ce3d030107
  label:     apache_ecc_privatekey
  ID:       123456
  Usage:    verify, derive
```


[9.3] GENERATE A CERTIFICATE SIGNING REQUEST (CSR) USING THE APACHE SERVER PRIVATE KEY

1. In a terminal, run the following command to generate a CSR from the private key that pkcs11-tool created on the Vectera Plus for Apache Server in the previous step:

```
$ sudo openssl req -new -engine pkcs11 -keyform engine -key "pkcs11:object=apache_ecc_privatekey" -out apache-cert-req.pem
```

Note: The common name of the Apache Server certificate should match the domain name or IP of the virtual host that you are configuring it for.

[9.4] CREATE A SELF-SIGNED ROOT CERTIFICATE AUTHORITY (CA)

Note: Here we are creating and using a self-signed root certificate authority (CA) for demonstration purposes. In a production environment, use a secure certificate authority (such as the KMES Series 3) should be used for all private key generation and certificate signing operations.

1. In a terminal, run the following commands to generate a root private key and self-signed certificate. We will use this certificate to sign the Apache Server certificate in the next section.

```
$ sudo openssl genrsa -out ssl-ca-privatekey.pem 2048
$ sudo openssl req -new -x509 -key ssl-ca-privatekey.pem -out ssl-ca-cert.pem -days 365
```

[9.5] SIGN THE APACHE SERVER CSR

1. In a terminal, run the following command to issue a signed Apache Server certificate using the self-signed root CA created in the previous step:

```
$ sudo openssl x509 -req -in apache-cert-req.pem -CA ssl-ca-cert.pem -CAkey ssl-ca-privatekey.pem -CAcreateserial -days 365 -out signed-apache-cert.pem
```

[9.6] CONFIGURE APACHE TO USE THE SIGNED CERTIFICATE AND PRIVATE KEY STORED ON THE HSM

This section covers how to modify the configuration file for an Apache virtual host. Configuration of a virtual host is outside the scope of this guide. Please reference the following [documentation](#) specific to your operating system if you do not already have a virtual host configured.

1. In a text editor, open the configuration file for the virtual host you want to configure HTTPS for and modify it as shown below:

```
<IfModule mod_ssl.c>
  <VirtualHost _default_:443>

    ServerAdmin webmaster@localhost
    ServerName myserver.local
    DocumentRoot /var/www/myserver.local
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
    SSLEngine on
    SSLCertificateFile      /etc/apache2/ssl/signed-apache-cert.pem
    SSLCertificateKeyFile  "pkcs11:object=apache_ecc_privatekey;type=private"

    <FilesMatch "\.(?:cgi|shtml|phtml|php)$">
      SSLOptions +StdEnvVars
    </FilesMatch>
    <Directory /usr/lib/cgi-bin>
      SSLOptions +StdEnvVars
    </Directory>

  </VirtualHost>
</IfModule>
```

Note: The location of the signed Apache certificate specified in the **SSLCertificateFile** define needs to be modified according to where it is stored on your system. The object name of the Apache private key specified in the **SSLCertificateKeyFile** define must match the label you set in the pkcs11-tool command in section 9.2.

2. Restart Apache to save and apply the configuration.

[9.7] CREATE A CLIENT CERTIFICATE FOR THE BROWSER THAT CONNECTS TO APACHE HTTP SERVER

Note: This step is only required if you want to use mutual authentication between the browser and the Apache HTTP Server.

1. In a terminal, generate a client keypair using the following command:

```
$ sudo openssl genrsa -out ssl-client-privatekey.pem 2048
```

2. Create a client certificate signing request:

```
$ sudo openssl req -new -key ssl-client-privatekey.pem -out ssl-client-req.pem -days 365
```

3. Sign the CSR with the CA certificate you created in section 9.4:

```
$ sudo openssl x509 -req -in ssl-client-req.pem -CA ssl-ca-cert.pem -CAkey ssl-ca-privatekey.pem -CAcreateserial -days 365 -out ssl-client-cert.pem
```

4. Convert the signed client certificate to PKCS #12 format for insertion into the browser:

```
$ sudo openssl pkcs12 -inkey ssl-client-privatekey.pem -in ssl-client-cert.pem -CAfile ssl-ca-cert.pem -export -out ssl-client-pkcs12.p12
```

[9.8] CONFIRM THAT APACHE USES THE NEW TLS CERTIFICATE AND PRIVATE KEY (STORED ON THE HSM) FOR HTTPS CONNECTIONS

If you did not create a client certificate in the previous section for mutual authentication, skip to step 4 below. You can complete following steps with a Firefox web browser. There may be some differences in the actions taken when using a different browser, but the overall intent of the process will be the same.

1. In Firefox, select **Settings > Privacy & Security > Certificates > View Certificates**.
2. Select **Your Certificates > Import** to import the PKCS #12 client certificate (i.e., ssl-client-pkcs12.p12).
3. Select **Authorities > Import** to import the CA certificate (ssl-ca-cert.pem).
4. Browse to the IP address of Apache website that is running over HTTPS. If you configured a client certificate in the browser for mutual authentication, you should see a lock icon next to the web address. If you did not configure a client certificate, bypass the warning and connect to the website.
5. View the certificate that the website served to the browser and confirm that it is the certificate configured in Apache in section 9.6.

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

EXCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

SOLUTIONS ARCHITECT

E-mail: solutions@futurex.com