# APACHE HTTP SERVER

Integration Guide

**Applicable Devices:**

*KMES Series 3*

**Applicable Versions:**

*6.3.1.x*

## 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 KMES Series 3 with Apache HTTP Server using PKCS #11 libraries. For additional questions related to your KMES Series 3 device, see the relevant user guide.

## [1.2] APPLICATION DESCRIPTION

### [1.2.1] About Apache HTTP Server

Apache HTTP Server, typically referred to as simply "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 webservers in the world. Apache is developed and maintained by an open community of developers under the auspices 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. While there's a lot of complexity underpinning how a web server functions, the primary job of all web servers is to accept requests from clients (e.g., a visitor's web browser) and then send the response to that request (e.g., the components of the page that a visitor wants to see).

### [1.2.3] Using HSMs to protect Apache Server private keys

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 via the HTTPS protocol, the Apache HTTP server uses the OpenSSL library. OpenSSL does not support PKCS #11 natively. To utilize HSMs, you have to 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 a certificate in the configuration file for the appropriate website.

## [2] PREREQUISITES

**Supported Hardware:**

- KMES Series 3, 6.3.1.x and above

**Supported Operating Systems:**

- Linux
- Windows 7 and above

**Other:**

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

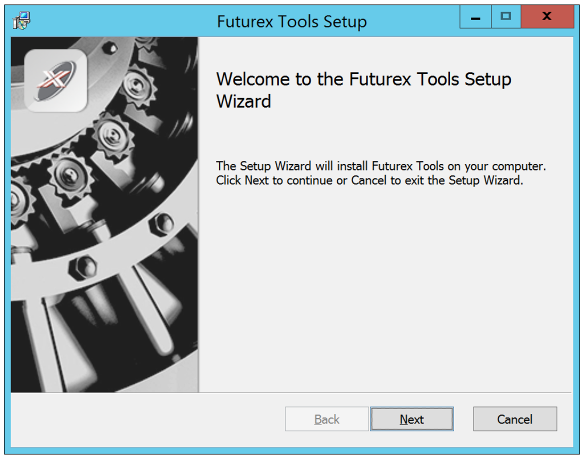
## [3] INSTALL FUTUREX PKCS #11 (FXPKCS11)

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

**NOTE:** The Futurex PKCS #11 module needs to be installed on the computer/server where Apache HTTP Server will be installed.

### [3.1] INSTRUCTIONS FOR INSTALLING THE FXPKCS11 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.

**NOTE:** Only the HSM version of the PKCS #11 configuration file is installed. For KMES integrations, the `<HSM>` section needs to be replaced with a `<KMS>` section.

## [3.2] INSTRUCTIONS FOR INSTALLING THE FXPKCS11 MODULE IN LINUX

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

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

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

- *fxpkcs11.cfg* -> PKCS #11 configuration file to use for HSM integrations
- *fxpkcs11-kms.cfg* -> PKCS #11 configuration file to use for KMES Series 3 integrations
- *x86/* - This folder contains the module files for 32-bit architecture
- *x64/* - This folder contains the module files for 64-bit architecture

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

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

The *configTest* and *PKCS11Manager* programs look for the PKCS #11 configuration file in the */etc* directory. Because of this, it is necessary either to move the PKCS #11 configuration file from the */usr/local/bin/fxpkcs11* directory to the */etc* directory, or to set the FXPKCS11_CFG environment variable to point to the PKCS #11 configuration file.

**NOTE:** If using the KMES version of the PKCS #11 configuration file (i.e., *fxpkcs11-kms.cfg*), the file needs to be renamed to *fxpkcs11.cfg*.

## [4] KMES SERIES 3 CONFIGURATION

The first half of this section will cover general KMES configurations that need to be made to allow Apache HTTP Server to integrate with the KMES to store the private key used for HTTPS connections. The second half of this section will cover the steps needed to configure TLS communication between the KMES Series 3 and the Futurex PKCS #11 library, which Apache will be utilizing to communicate with the KMES.

### [4.1] CREATE A ROLE AND IDENTITY FOR APACHE

A new role and identity need to be created on the KMES Series 3. This role will be assigned to the identity and subsequently will be used by the Futurex PKCS #11 library to connect to the KMES Series 3.

1. Log in to the KMES Series 3 application interface with the default Admin identities.
2. Navigate to the *Identity Management* -> *Roles* menu and click the **Add** button. This will pull up the *Role Editor* dialog.
3. Specify a name for the role, select the "Hardened" checkbox, and set the number of logins required to "1".
4. In the *Permissions* tab, select the following permissions:
    - **Certificate Authority** -> **Add**,**Export**
    - **Cryptographic Operations** -> **Sign**
    - **Keys** -> **Add**
5. In the *Advanced* tab, set the Allowed Ports field to **Host API**.
6. Click the **OK** button to finish creating the role.
7. Navigate to the *Identity Management* -> *Identities* menu, then right-click and select **Add** -> **Client Application**.
8. Change the storage type to "HSM" and specify a name for the identity.
9. In the *Assigned Roles* tab, select the hardened role that was just created.
10. In the *Authentication* tab, click the **Add** button to configure a new credential. This will pull up the *Configure Credential* dialog.
11. Set the credential type to **Password** and set a password for the credential, then click **OK**. The new Password credential should be listed now with the API Key credential that exists by default.
12. Select the API Key credential and click **Remove**.
13. In the main *Identity Editor* dialog, click **OK** to save. The new identity should now be listed along with the other identities that exist on the KMES Series 3.

## [4.2] ENABLE THE HOST API COMMANDS REQUIRED FOR THE APACHE HTTP SERVER OPERATION

Because the Futurex PKCS #11 library will be connecting to the Host API port on the KMES, users must define which Host API commands will be enabled for execution by the FXPKCS11 library. To set the enabled commands, complete the following steps:

1. Go to *Administration -> Configuration -> Host API Options*, enable the commands listed below, then click **Save**.

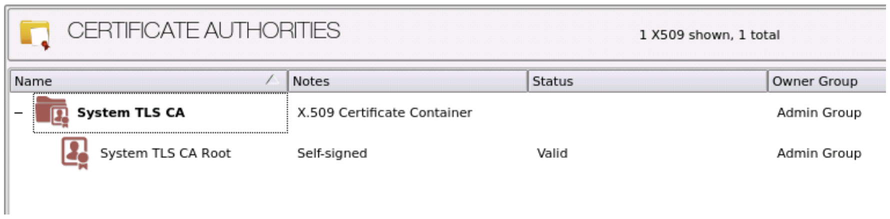
- **ATKG**: Manipulate HSM trusted asymmetric key group
    - **add**: Add HSM trusted asymmetric key group
    - **modify**: Modify HSM trusted asymmetric key group
    - **delete**: Delete HSM trusted asymmetric key group
    - **get**: Retrieve HSM trusted asymmetric key group
- **ECHO**: Communication Test/Retrieve Version
- **RAFA**: Filter Issuance Policy
- **RAND**: Generate Random Number
- **RKCK**: Create HSM Trusted Key
- **RKCP**: Get Command Permissions
- **RKCS**: Create Symmetric HSM Trusted Key Group
- **RKGP**: Export Asymmetric HSM Trusted Key
- **RKGS**: Generate Signature
- **RKLN**: Lookup Objects
- **RKLO**: Login User
- **RKPK**: Pop Generated Key
- **TIME**: Set Time

## [4.3] CONFIGURE TLS COMMUNICATION BETWEEN THE KMES SERIES 3 AND THE FUTUREX PKCS #11 (FXPKCS11) LIBRARY

### [4.3.1] Create a Certificate Authority (CA)

1. Navigate to the *PKI -> Certificate Authorities* menu, then click the **Add CA...** button at the bottom of the page.
2. In the *Certificate Authority* dialog, enter a name for the Certificate Container, leave all other fields as the default values, then click **OK**.
3. Right-click on the Certificate Container that was created and select **Add Certificate -> New Certificate...**

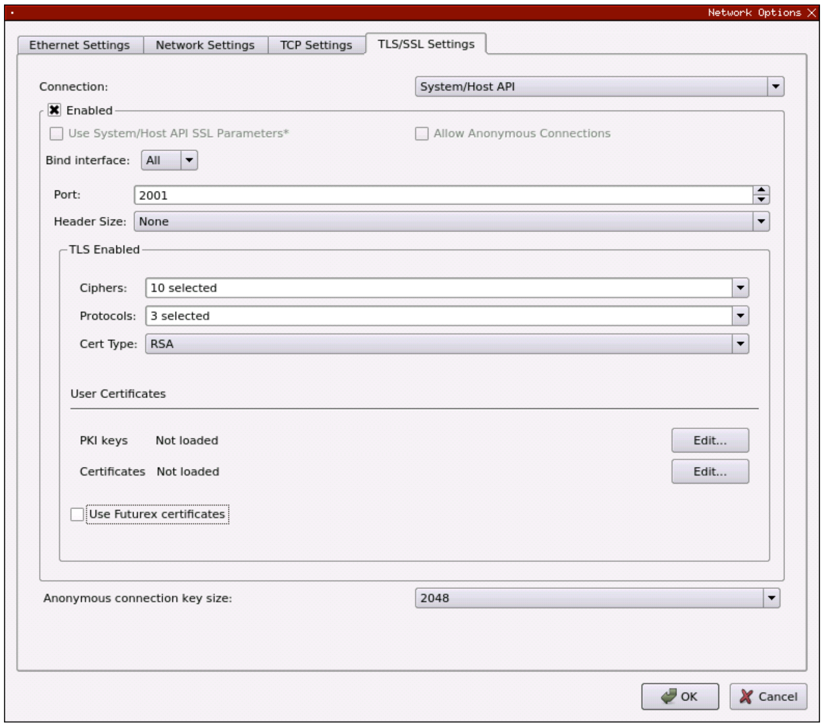
4. In the *Subject DN* tab, select the "Classic" preset and set a Common Name for the certificate, such as "System TLS CA Root".
5. In the *Basic Info* tab, leave all settings as the default values.
6. In the *V3 Extensions* tab, select the "Certificate Authority" profile, then click **OK**. The root CA certificate will be listed now under the previously created Certificate Container.



| Name | Notes | Status | Owner Group |
|---|---|---|---|
| System TLS CA | X.509 Certificate Container | | Admin Group |
| System TLS CA Root | Self-signed | Valid | Admin Group |

## [4.3.2] Generate a CSR for the System/Host API connection pair

1. Go to *Administration -> Configuration -> Network Options*.
2. In the *Network Options* dialog, select the *TLS/SSL Settings* tab.
3. Under the **System/Host API** connection pair, uncheck "Use Futurex certificates", then click **Edit...** next to PKI keys in the User Certificates section.



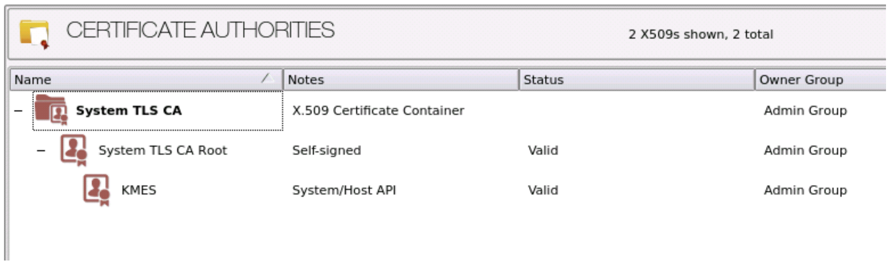
Connection: System/Host API
[✓] Enabled
[ ] Use System/Host API SSL Parameters*
[ ] Allow Anonymous Connections
Bind interface: All
Port: 2001
Header Size: None
TLS Enabled
Ciphers: 10 selected
Protocols: 3 selected
Cert Type: RSA
User Certificates
PKI keys Not loaded
Certificates Not loaded
[ ] Use Futurex certificates
Anonymous connection key size: 2048

4. In the *Application Public Keys* dialog, click **Generate...**
5. There will be a warning stating that SSL will not be functional until new certificates are imported. Select **Yes** if you wish to continue.
6. In the *PKI Parameters* dialog, leave the default settings and click **OK**.

7. It should show that a PKI Key Pair is loaded now in the *Application Public Keys* dialog. If this is the case, click **Request...**
8. In the *Subject DN* tab, set a Common Name for the certificate, such as "KMES".
9. In the *V3 Extensions* tab, select the "TLS Server Certificate" profile.
10. In the *PKCS #10 Info* tab, select a save location for the CSR, then click **OK**.
11. There should be a message stating that the certificate signing request was successfully written to the file location that was selected. Click **OK**.
12. Click **OK** again to save the *Application Public Keys* settings.
13. In the main *Network Options* dialog, it should now say "Loaded" next to **PKI keys** for the System/Host API connection pair.

## [4.3.3] Sign the System/Host API CSR

1. Navigate to the *PKI* -> *Certificate Authorities* menu.
2. Right-click on the "System TLS CA Root" certificate created in section 4.3.1, then select **Add Certificate** -> **From Request...**
3. In the file browser, find and select the CSR that was generated for the System/Host API connection pair.
4. Once loaded, none of the settings need to be modified for the certificate. Click **OK**.
5. The signed System/Host API TLS certificate should now show under the TLS root CA certificate on the *Certificate Authorities* page.


CERTIFICATE AUTHORITIES

2 X509s shown, 2 total

| Name | Notes | Status | Owner Group |
|---|---|---|---|
| – System TLS CA | X.509 Certificate Container | | Admin Group |
| – System TLS CA Root | Self-signed | Valid | Admin Group |
| KMES | System/Host API | Valid | Admin Group |

## [4.3.4] Export the TLS Root CA certificate

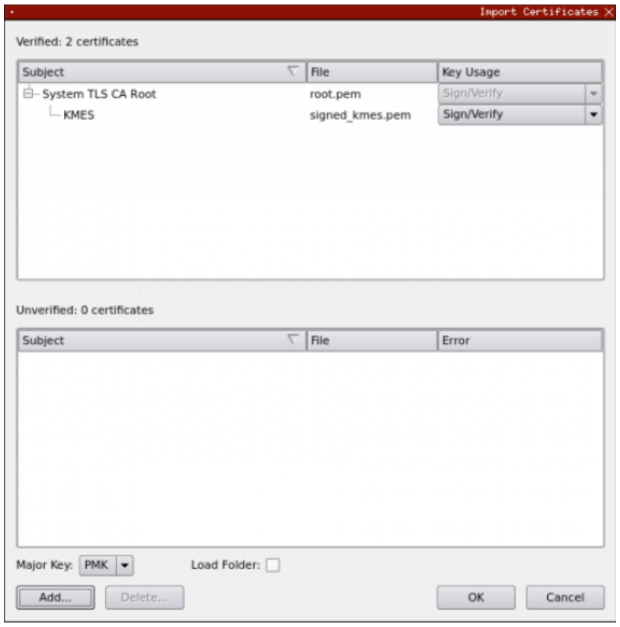
1. Navigate to the *PKI* -> *Certificate Authorities* menu.
2. Right-click on the "System TLS CA Root" certificate, then select **Export** -> **Certificate(s)...**
3. In the *Export Certificate* dialog, change the encoding to "PEM", then click **Browse...**
4. In the file browser, navigate to the location where you want to save the TLS root CA certificate. Specify a name for the file, then click **Open**.
5. Click **OK**. A message box will pop up stating that the PEM file was successfully written to the location that you specified.

## [4.3.5] Export the signed System/Host API TLS certificate

1. Navigate to the *PKI -> Certificate Authorities* menu.
2. Right-click on the "KMES" certificate, then select **Export** -> **Certificate(s)...**
3. In the *Export Certificate* dialog, change the encoding to "PEM", then click **Browse...**
4. In the file browser, navigate to the location where you want to save the signed System/Host API TLS certificate. Specify a name for the file, then click **Open**.
5. Click **OK**. A message box will pop up stating that the PEM file was successfully written to the location that you specified.

## [4.3.6] Load the exported TLS certificates into the System/Host API connection pair

1. Go to *Administration -> Configuration -> Network Options*.
2. In the *Network Options* dialog, select the *TLS/SSL Settings* tab.
3. Click **Edit...** next to Certificates in the User Certificates section.
4. Right-click on the **System/Host API SSL CA** X.509 Certificate Container, then select **Import...**
5. Click **Add...** at the bottom of the *Import Certificates* dialog.
6. In the file browser, find and select both the TLS root CA certificate and the signed System/Host API TLS certificate, then click **Open**. The certificate chain should appear in the "Verified" section, as shown below:



Verified: 2 certificates

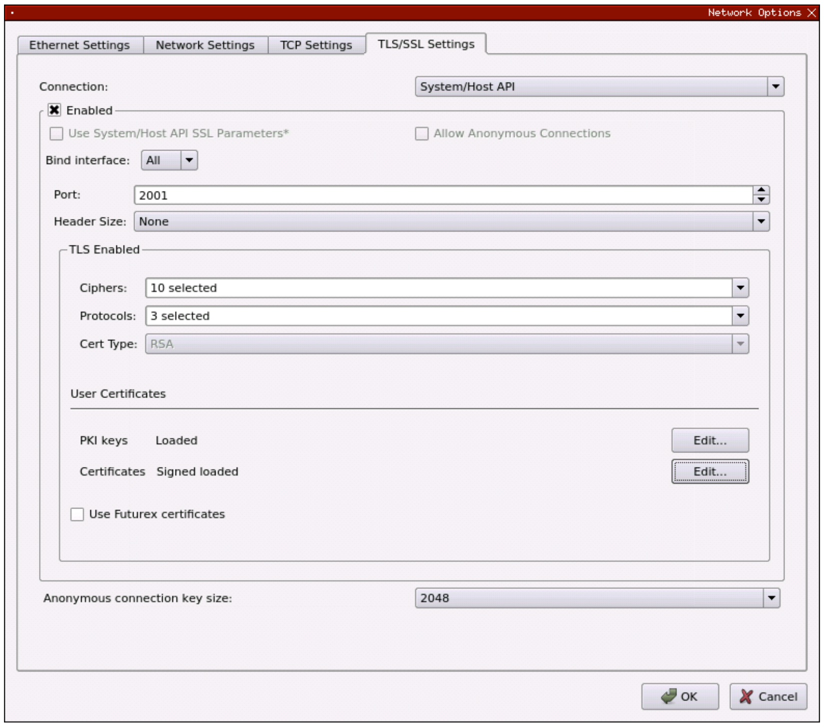
| Subject | File | Key Usage |
|---|---|---|
| System TLS CA Root | root.pem | Sign/Verify |
| KMES | signed_kmes.pem | Sign/Verify |

Unverified: 0 certificates

| Subject | File | Error |
|---|---|---|

Major Key: PMK    Load Folder:

7. Click **OK** to save the changes. In the *Network Options* dialog, the System/Host API connection pair should show "Signed loaded" next to Certificates in the User Certificates section, as shown below:



Connection: System/Host API
[✓] Enabled
[ ] Use System/Host API SSL Parameters*
[ ] Allow Anonymous Connections
Bind interface: All
Port: 2001
Header Size: None
TLS Enabled
Ciphers: 10 selected
Protocols: 3 selected
Cert Type: RSA
User Certificates
PKI keys Loaded
Certificates Signed loaded
[ ] Use Futurex certificates
Anonymous connection key size: 2048

8. Click **OK** to save and exit the Network Options dialog.

## [4.3.7] Generate a TLS private key and certificate signing request for the Futurex PKCS #11 (FXPKCS11) library using OpenSSL

**NOTE:** The commands in this section need to be run from a terminal application that has OpenSSL installed.

1. Open a terminal and run the following command to generate a TLS private key for the FXPKCS11 library:

```
$ openssl genrsa -out fxpkcs11_tls_privatekey.pem 2048
```

   The private key will be output to a file named `fxpkcs11_tls_privatekey.pem` in the same directory that the command was run from.

2. Run the following command to generate a Certificate Signing Request (CSR) for the FXPKCS11 library:

```
$ openssl req -new -key fxpkcs11_tls_privatekey.pem -out fxpkcs11_tls_cert_req.pem -days 365
```

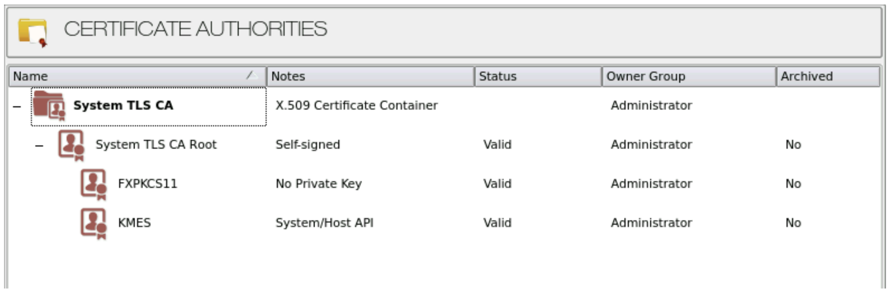
   It will prompt you to enter certificate information. Set the default value for every field by pressing **Enter** at every prompt.

   The CSR will be output to a file named `fxpkcs11_tls_cert_req.pem` in the same directory that the command was run from.

3. Move or copy the CSR file (`fxpkcs11_tls_cert_req.pem`) to the storage medium configured on the KMES.

## [4.3.8] Sign the Certificate Signing Request (CSR) for the FXPKCS11 library

1. Navigate to the *PKI* -> *Certificate Authorities* menu.
2. Right-click on the "System TLS CA Root" certificate and select **Add Certificate** -> **From Request...**
3. In the file browser, find and select the FXPKCS11 CSR. Certificate information will populate in the *Create X.509 From CSR* window.
4. In the *Subject DN* tab, change the preset dropdown to "Classic", then set a Common Name for the certificate, such as "FXPKCS11".
5. All settings in the *Basic Info* tab can be left as the default values.
6. In the *V3 Extensions* tab, select the "TLS Client Certificate" profile, then click **OK**.
7. The signed "FXPKCS11" certificate will be listed now under the TLS root certificate.


CERTIFICATE AUTHORITIES

| Name | Notes | Status | Owner Group | Archived |
|---|---|---|---|---|
| – **System TLS CA** | X.509 Certificate Container | | Administrator | |
| – System TLS CA Root | Self-signed | Valid | Administrator | No |
| FXPKCS11 | No Private Key | Valid | Administrator | No |
| KMES | System/Host API | Valid | Administrator | No |

## [4.3.9] Export the signed FXPKCS11 TLS certificate

1. Navigate to the *PKI* -> *Certificate Authorities* menu.
2. Right-click on the "FXPKCS11" certificate, then select **Export** -> **Certificate(s)...**
3. In the *Export Certificate* dialog, change the encoding to "PEM", then click **Browse...**
4. In the file browser, navigate to the location where you want to save the FXPKCS11 TLS certificate. Specify a name for the file, then click **Open**.
5. Click **OK**. A message box will pop up stating that the PEM file was successfully written to the location that you specified.

**NOTE:** The signed FXPKCS11 TLS certificate and the TLS Root CA certificate that was exported in section 4.3.4 both need to be moved to the computer that will be running the Apache HTTP Server instance. In the next section, they will be configured and used for TLS communication with the KMES Series 3.

# [5] EDIT THE FUTUREX PKCS #11 CONFIGURATION FILE

## [5.1] DEFINE CONNECTION INFORMATION

The *fxpkcs11.cfg* file allows the user to set the FXPKCS11 library to connect to the KMES Series 3. To edit, run a text editor as an Administrator on Windows or as root on Linux, and edit the configuration file accordingly. Most notably, the fields shown below must be set inside the **<KMS>** section (note that the entire *fxpkcs11.cfg* file is not included).

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

```
<KMS>
    # Which PKCS11 slot
    <SLOT>                  0                       </SLOT>

    # Login username
    <CRYPTO-OPR>            crypto1                 </CRYPTO-OPR>

    # Key group name
    #<KEYGROUP-NAME>        keygroup1               </KEYGROUP-NAME>

    # Asymmetric key group name
    <ASYM-KEYGROUP-NAME>    asymkeygroup1           </ASYM-KEYGROUP-NAME>

    # Connection information
    <ADDRESS>               10.0.8.20 </ADDRESS>
    <PROD-PORT>             2001                    </PROD-PORT>
    <PROD-TLS-ENABLED>      YES                     </PROD-TLS-ENABLED>
    <PROD-TLS-ANONYMOUS>    NO                      </PROD-TLS-ANONYMOUS>
    <PROD-TLS-CA>           /connection_certs/root_tls_cert.pem        </PROD-TLS-CA>
    <PROD-TLS-CERT>         /connection_certs/signed_fxpkcs11_tls_cert.pem      </PROD-TLS-CERT>
    <PROD-TLS-KEY>          /connection_certs/fxpkcs11_tls_privatekey.pem   </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>
</KMS>
```

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

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

The **<KEYGROUP-NAME>** field can be used when an application needs to create symmetric keys on the KMES. For this integration, this field can be commented out because Apache only needs to create an asymmetric key pair on the KMES.

The **<ASYM-KEYGROUP-NAME>** field does need to be defined for this integration. The asymmetric key that Apache creates on the KMES will be added to a key group with the name specified here.

In the **<ADDRESS>** field, specify the IP of the KMES that the PKCS #11 library should connect to.

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

In the **<PROD-PORT>** field, set the PKCS #11 library to connect to the default Host API port on the KMES, port 2001.

The **<PROD-TLS-ENABLED>** field needs to be set to "YES" because the only way to connect to the Host API port on the KMES is over TLS.

The **<PROD-TLS-ANONYMOUS>** field defines whether the PKCS #11 library will be authenticating to the KMES or not. Since we're connecting to the Host API port using mutual authentication, 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. In this example, there is only one CA certificate.

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.

If a Guardian is being used to manage KMES Series 3 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”.

Once the *fxpkcs11.cfg* file is edited, run the *PKCS11Manager* file to test the connection against the KMES, 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.

## [5.2] SPECIAL DEFINE REQUIRED FOR THIS INTEGRATION

The following define must be added to the **<CONFIG>** section of the FXPKCS11 configuration file:

```
<FORCED-ASYMMETRIC-USAGE>   SIGN | VERIFY       </FORCED-ASYMMETRIC-USAGE>
```

# [6] OPENSSL ENGINE INSTALLATION AND CONFIGURATION

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

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

## [6.1] 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
```

## [6.2] EDIT THE OPENSSL CONFIGURATION FILE

**NOTE:** The following instructions are the same for Ubuntu/Debian-based Linux distributions and Red Hat/CentOS-based distributions, with the exception of the `dynamic_path` define in the *openssl.cnf* file. On Ubuntu 18, the location of the *pkcs11.so* file specified in the `dynamic_path` define is as shown below. The location of the *pkcs11.so* file on CentOS 8 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 logged in user:

```
openssl version -d
```

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

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

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

   ```
   openssl_conf = openssl_init
   ```

   This needs be added to the bottom of the file:

   ```
   [openssl_init]
   engines=engine_section
   [engine_section]
   pkcs11 = pkcs11_section
   [pkcs11_section]
   engine_id = pkcs11
   dynamic_path = /usr/lib/x86_64-linux-gnu/engines-1.1/pkcs11.so
   MODULE_PATH = /usr/local/bin/fxpkcs11/libfxpkcs11.so
   PIN = "safest"
   init = 0
   ```

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

   **NOTE:** The password of the identity created on the KMES Series 3 needs to be set in the PIN field.

# [7] APACHE HTTP SERVER CONFIGURATION

## [7.1] SET FXPKCS11 ENVIRONMENT VARIABLES

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

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

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

## [7.2] CREATE A KEY PAIR ON THE KMES SERIES 3 USING PKCS11-TOOL

In a terminal, run the following command to create a new key pair on the KMES Series 3 using pkcs11-tool:

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

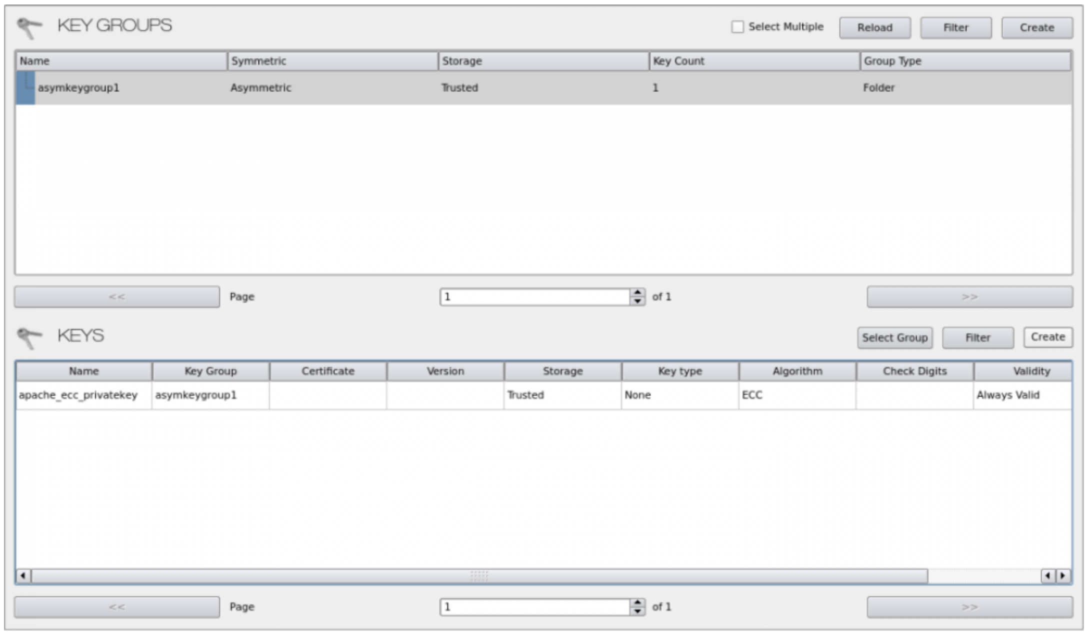
**NOTE:** At the time of writing, there is a bug in Apache that prevents RSA certificates from being served correctly to the browser. This bug may be fixed, but to be safe it is recommended to create and use an ECC certificate as demonstrated.

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

```
Key pair generated:
Private Key Object; EC
  label:      apache_ecc_privatekey
  ID:         123456
  Usage:      sign
Public Key Object; EC  EC_POINT 256 bits
  EC_POINT:   04410455ff9a32b8c9734c-
c2d37825a009916-
abf09f053e3b6b1a2c4ce2e0f87fa2a2a76b4bf82b3fce388c4804c3d031cc343006ef6ff80acf6bd72ae2044d1be5efd
  EC_PARAMS:  06082a8648ce3d030107
  label:      apache_ecc_privatekey
  ID:         123456
  Usage:      verify
```

One private ECC 256-bit key was created with asymmetric sign usage, and one public ECC 256-bit key was created with verify usage.

In the KMES application interface, we can see that a new asymmetric key group was created (using the value specified with the `<ASYM-KEYGROUP-NAME>` tag in the *fxpkcs11.cfg* file), and inside the key group one key pair was created with the name specified in the above **pkcs11-tool** command.



| Name | Symmetric | Storage | Key Count | Group Type |
|---|---|---|---|---|
| asymkeygroup1 | Asymmetric | Trusted | 1 | Folder |

| Name | Key Group | Certificate | Version | Storage | Key type | Algorithm | Check Digits | Validity |
|---|---|---|---|---|---|---|---|---|
| apache_ecc_privatekey | asymkeygroup1 | | | Trusted | None | ECC | | Always Valid |

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

**NOTE:** Before completing the remaining steps in this section, create a directory to store the TLS certificates that will be created, then navigate to that directory.

In a terminal, run the following command to generate a CSR using the private key that was created on the KMES Series 3 for Apache Server:

```
$ 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 of the virtual host it will be configured for.

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

**NOTE:** A self-signed root certificate authority (CA) is being used here for demonstration purposes. In a production environment, a secure certificate authority (such as the KMES Series 3) should be used for all private key generation and certificate signing operations.

In a terminal, run the following sequence of commands to generate a root private key and self-signed certificate. This certificate will be used 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
```

## [7.5] SIGN THE APACHE SERVER CSR

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
```

## [7.6] CONFIGURE APACHE TO USE THE SIGNED CERTIFICATE AND THE PRIVATE KEY STORED ON THE KMES SERIES 3

This section will cover how to modify the configuration file for a virtual host that is running in Apache. Configuring 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 that you want to configure HTTPS for and modify it as shown below.

   **NOTE:** The location of the configuration file will specific to your system.

   ```
   <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.

   **NOTE:** The object name of the Apache private key specified in the `SSLCertificateKeyFile` define needs to match the label that was set in the pkcs11-tool command in section 7.2.

2. Restart Apache to save and apply the configuration.

## [7.7] CREATE A CLIENT CERTIFICATE FOR THE BROWSER THAT WILL BE CONNECTING TO APACHE HTTP SERVER

**NOTE:** This step is only required if you would like to use mutual authentication.

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 that was created in section 7.4:

```
$ 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
```

## [7.8] CONFIRM THAT APACHE IS USING THE NEW TLS CERTIFICATE AND PRIVATE KEY (STORED ON THE KMES) FOR HTTPS CONNECTIONS

**NOTE:** If a client certificate was not created for mutual authentication in the previous section, skip to step 4 below.

**NOTE:** The following steps were completed using 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, navigate to *Settings* -> *Privacy & Security* -> *Certificates* and click the **View Certificates** button.
2. Under the *Your Certificates* tab, select **Import** to import the client certificate that was converted to PKCS #12 (i.e., *ssl-client-pkcs12.p12*).
3. Under the *Authorities* tab, select **Import** to import the CA certificate (i.e., *ssl-ca-cert.pem*).
4. Navigate to the IP address from which Apache is running over HTTPS.

   **NOTE:** If a client certificate was configured in the browser for mutual authentication, you should see a lock icon next to the web address. If a client certificate was not configured, bypass the warning and connect to the website anyway.

5. View the certificate that the website served to the browser and confirm that it is the certificate that was configured in Apache in section 7.6.

## 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