SSH KEY OFFLOADING USING PKCS #11

Integration Guide

Applicable Devices:
KMES Series 3

TABLE OF CONTENTS

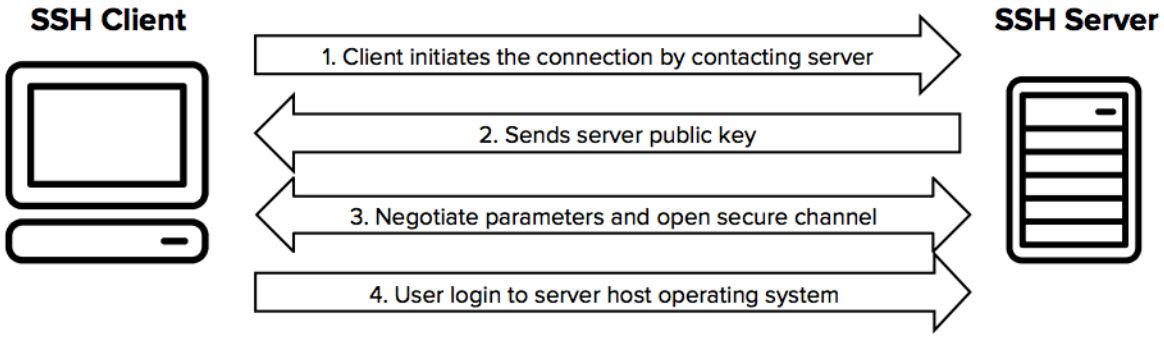
[1] OVERVIEW OF SSH KEY OFFLOADING TO THE KMES USING PKCS #11

[1.1] WHAT IS SSH?

SSH (Secure Shell) is a cryptographic network protocol that enables the ability to remotely log in from one computer to another in a secure manner.

[1.2] HOW DOES SSH WORK?

The SSH protocol uses a **client-server architecture**, which means that the connection is established by an **SSH client** connecting to an **SSH server**. The SSH client drives the connection setup process and uses public key cryptography to verify the SSH server's identity. After the setup phase, the SSH protocol uses strong symmetric encryption and hashing algorithms to ensure the privacy and integrity of the data that is exchanged between the client and server.



[1.3] SSH CLIENT AUTHENTICATION METHODS

Several methods can be used for SSH client authentication to the SSH server. Most commonly used are **password** and **public key** authentication.

[1.3.1] Password Authentication

With password authentication, an SSH client authenticates to an SSH server using the password of a user that exists on the SSH server. For example, an SSH client would attempt to establish an SSH connection using the following command:

```
ssh username@server.com
```

The SSH client would then be prompted for the password of the remote user it's attempting to connect with. If the password entered matches the password of the remote user on the SSH server, the remote login session will be established.

[1.3.2] Public Key Authentication

The public key authentication method is often preferred over password authentication because it is more secure and allows for increased automation. Once public key authentication is set up, the SSH client is no longer required to enter a password every time it connects.

Public key authentication can be set up with the following steps:

1. An SSH key pair (public key and private key) is generated on the SSH client.
2. The SSH client public key is then moved to the SSH server (i.e., via SCP or SFTP) and added to the `~/.ssh/authorized_keys` file.

Now, when the SSH client attempts to connect using the same command as before (`ssh username@server.com`) it should not be prompted for the remote user's password. Instead, this is what is what is happening "behind the scenes" during the connection:

1. The SSH client sends the following command to connect to the SSH server:
   ```
   ssh username@server.com
   ```
2. The SSH server checks its *`~/.ssh/authorized_keys`* file and finds a public key for the user that the SSH client is attempting to connect with.
3. The SSH server then asks the SSH client to sign some arbitrary data using its SSH client private key to prove that the SSH client is in possession of the private key corresponding to the public key.
4. The SSH client then sends data signed with the private key back to the SSH server, which then attempts to decrypt the data using the public key that it has for the user the SSH client is connecting with.
5. If the decryption is successful, the SSH server will then trust the SSH client, establishing authentication.

[1.4] HOW DOES THE KMES SERIES 3 FIT IN TO THE PROCESS?

By default, when an SSH key pair is created using the `ssh-keygen` command on an SSH client machine, the private key is stored in a plaintext file in the `~/.ssh` directory. This poses a security risk because any person with access to that machine can view the private key and use it to authenticate to remote machines over SSH.

Incorporating the KMES Series 3 into this process allows the SSH client private key to be stored within the confines of a FIPS 140-2 Level 3 validated hardware security module. The way that SSH is able to integrate with the KMES Series 3 is through the PKCS #11 library.

With the KMES Series 3 incorporated in this process, the SSH client would send the following command to connect to the SSH server, where *`FXPKCS11_MODULE_LOCATION`* is the location of the FXPKCS11 library file (`libfxpkcs11.so` on Linux and `fxpkcs11.dll` on Windows):

```
ssh -I [FXPKCS11_MODULE_LOCATION] client@server.com
```

Now when the SSH client needs to sign data using its private key (step 3-4 in the previous subsection), it is configured to point to the PKCS #11 library, which then automatically authenticates to the KMES series 3 with an identity and password set in the FXPKCS11 configuration file. The KMES Series 3 then signs some arbitrary data using the private key that corresponds to that identity and sends the signed data back to the SSH client, which forwards it to the SSH server.

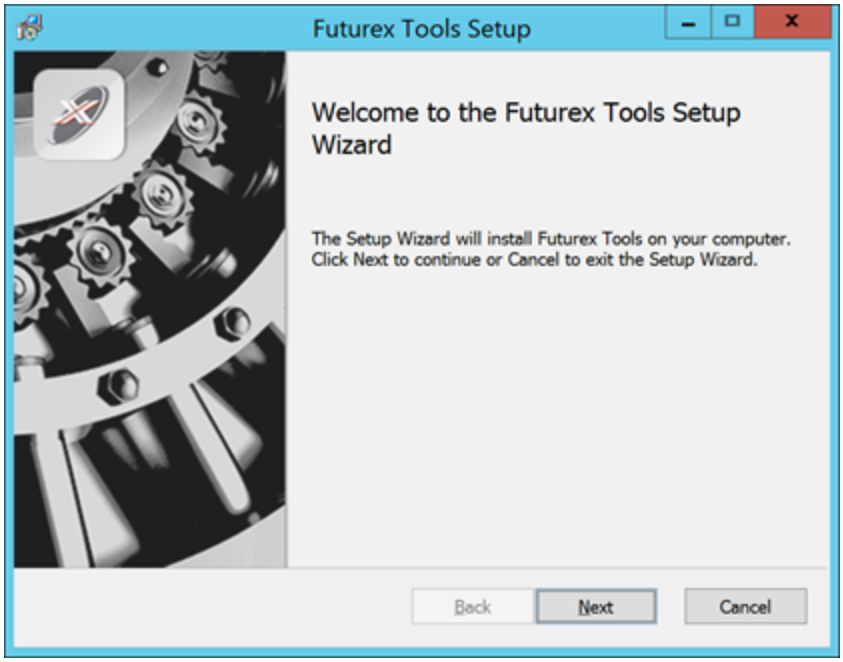
[2] INSTALL FUTUREX PKCS #11 (FXPKCS11)

NOTE: The steps in this section only need to be performed on the SSH client machine.

In a Windows environment, the easiest way to install the PKCS #11 module is by using 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.

[2.1] INSTRUCTIONS FOR INSTALLING THE PKCS #11 MODULE USING FXTOOLS IN WINDOWS

Run the FXTools installer as an administrator.



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

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

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

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

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

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

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

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

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

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

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

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

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

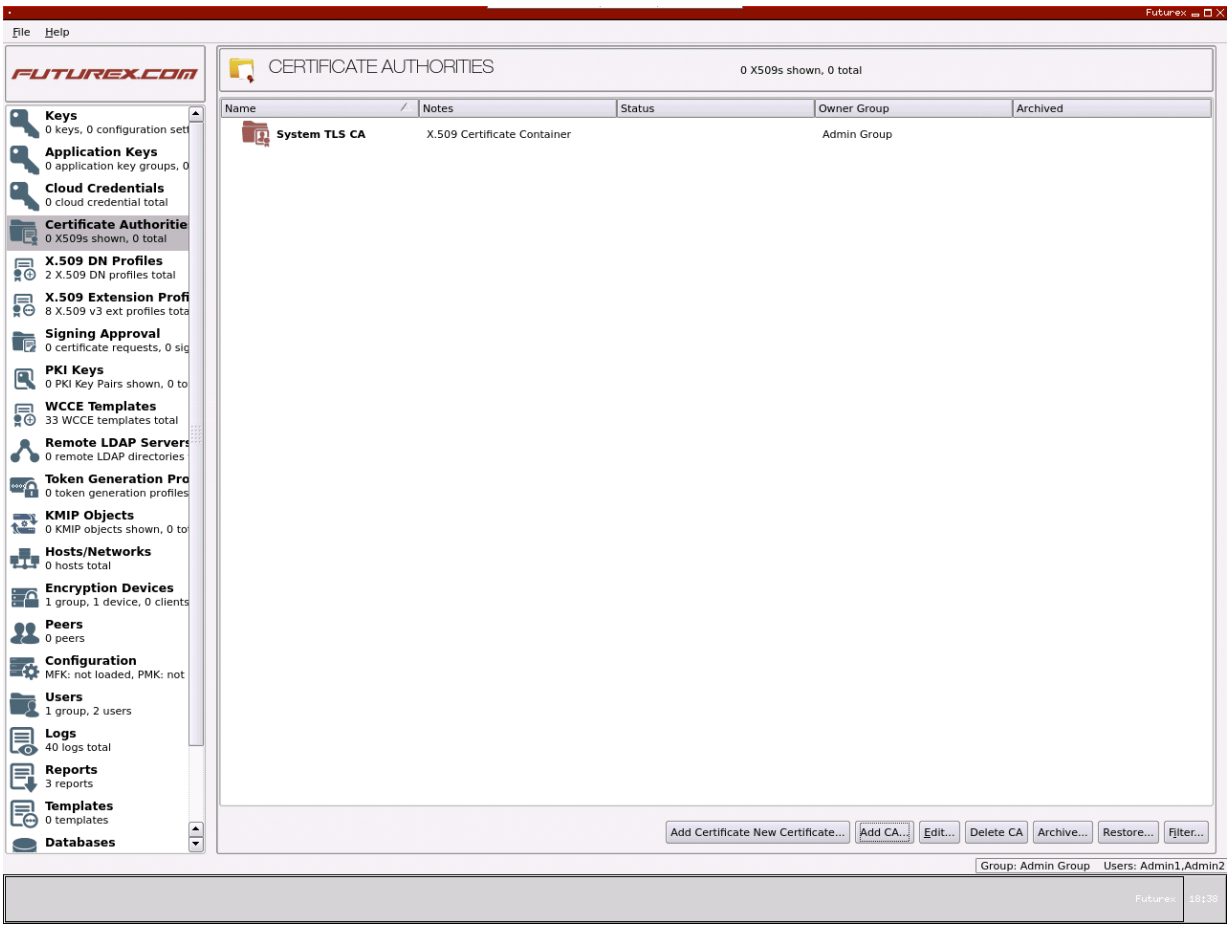
Because of this, it is necessary either to move the fxpkcs11.cfg file from the /usr/local/bin/fxpkcs11 directory to the /etc directory, or to set the FXPKCS11_CFG environment variable to point to the fxpkcs11.cfg file.

[3] KMES SERIES 3 CONFIGURATION

[3.1] CONFIGURE TLS COMMUNICATION BETWEEN THE KMES SERIES 3 AND THE SSH CLIENT PKCS #11 LIBRARY

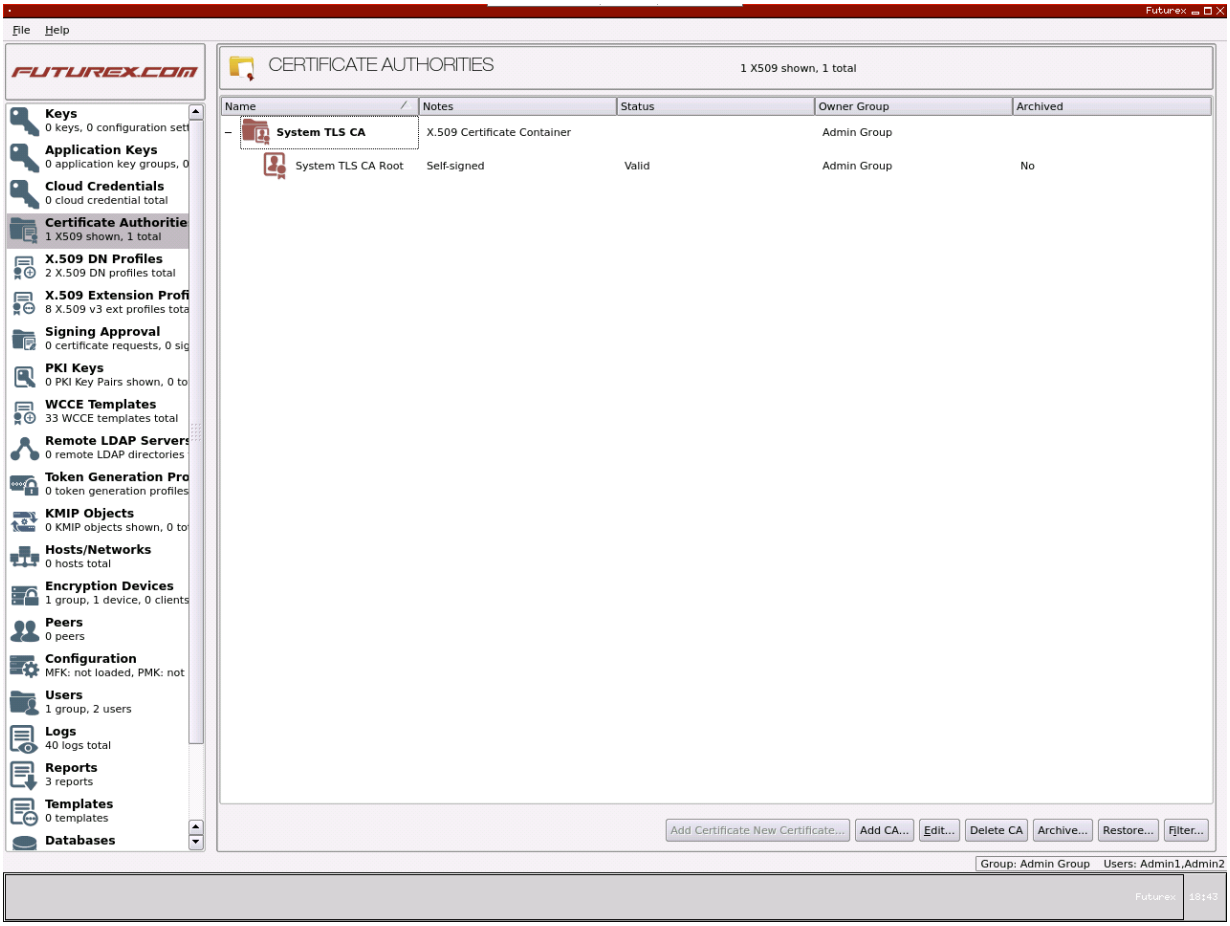
[3.1.1] Create a Certificate Authority (CA)

1. Log in to the KMES Series 3 application interface with the default Admin identities.
2. Select *Certificate Authorities* in the left menu, then click the **Add CA...** button at the bottom of the page.
3. In the *Certificate Authority* dialog, enter a name for the Certificate Container, leave all other fields as the default values, then click **OK**.
4. The Certificate Container that was just created will be listed now in the Certificate Authorities menu.



5. Right-click on the Certificate Container and select **Add Certificate -> New Certificate...**
6. In the *Subject DN* tab, set a Common Name for the certificate, such as "System TLS CA Root".

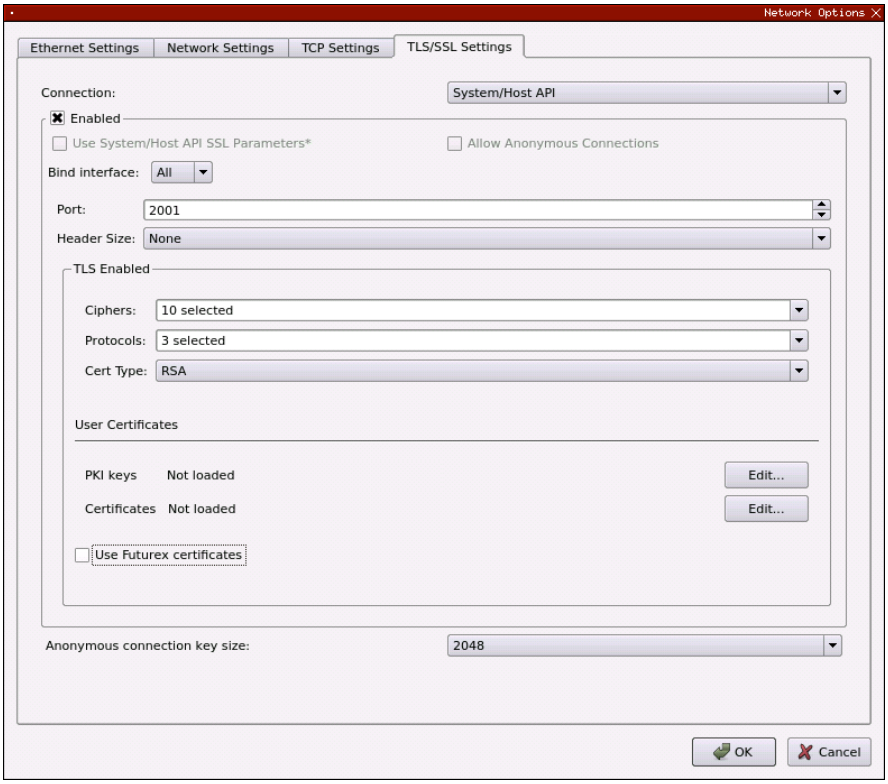
7. In the *Basic Info* tab, change the Major key to the **PMK**. All other settings can be left as the default values.
8. In the *V3 Extensions* tab, select the "Example Certificate Authority" profile, then click **OK**.
9. The root CA certificate will be listed now under the previously created Certificate Container.



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

1. Go to *Configuration -> Network Options*.
2. In the *Network Options* dialog, select the *TLS/SSL Settings* tab.

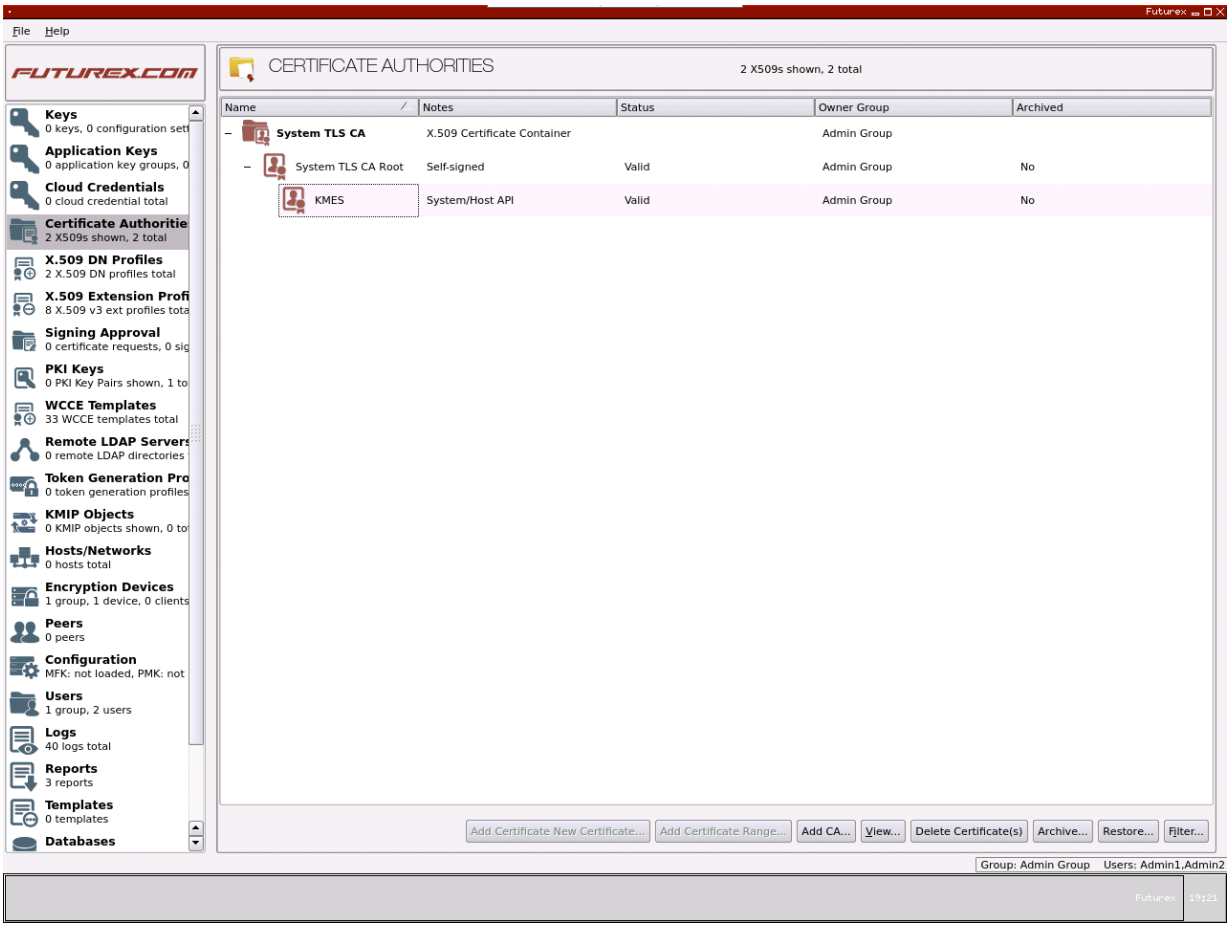
3. Under the **System/Host API** connection pair, uncheck "Use Futurex certificates", then click **Edit...** next to PKI keys in the User Certificates section.



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, change the Encrypting key to the **PMK**, then change the Key Size to **2048** 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 "Example 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.

[3.1.3] Sign the System/Host API CSR

1. Go to the *Certificate Authorities* menu.
2. Right-click on the root CA certificate created in section 2.1.1, 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 certificate should now show under the root CA certificate on the *Certificate Authorities* page.



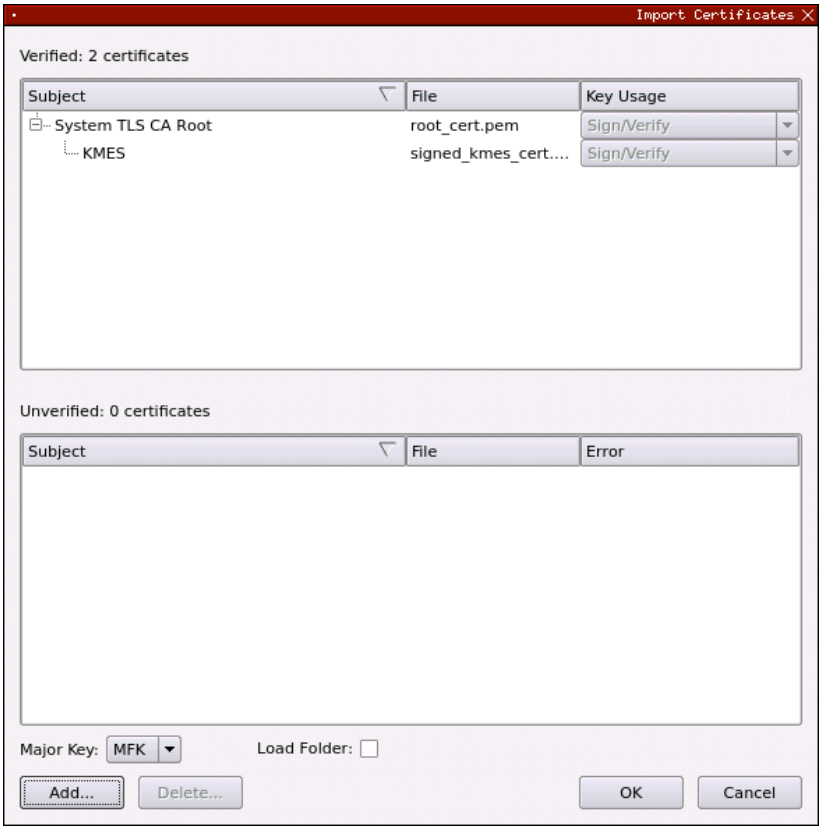
[3.1.4] Export the root CA and signed System/Host API certificates

1. Right-click on the root CA certificate, then select **Export** -> **Certificate(s)...**
2. Change the encoding to **PEM**. Then click **Browse...** and select a save location, as well as a name for the export file.
3. There should be a message stating that the file was successfully written to the location that was selected. Click **OK**.

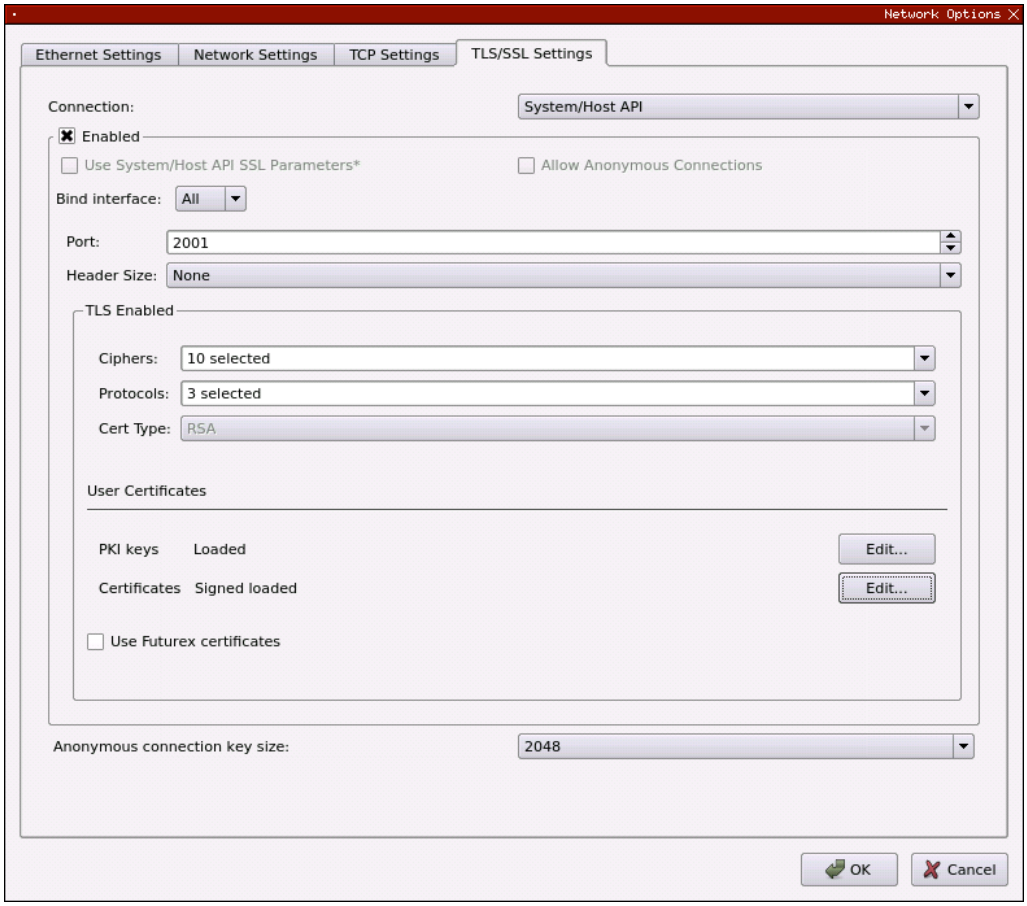
4. Right-click on the signed System/Host API certificate, then select **Export** -> **Certificate(s)...**
5. Change the encoding to **PEM**. Then click **Browse...** and select a save location, as well as a name for the export file.
6. There should be a message stating that the file was successfully written to the location that was selected. Click **OK**.

[3.1.5] Load the exported certificates into the System/Host API connection pair

1. Go to *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 root CA certificate and the signed System/Host API certificate, then click **Open**. The certificate chain should appear as shown below:



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:

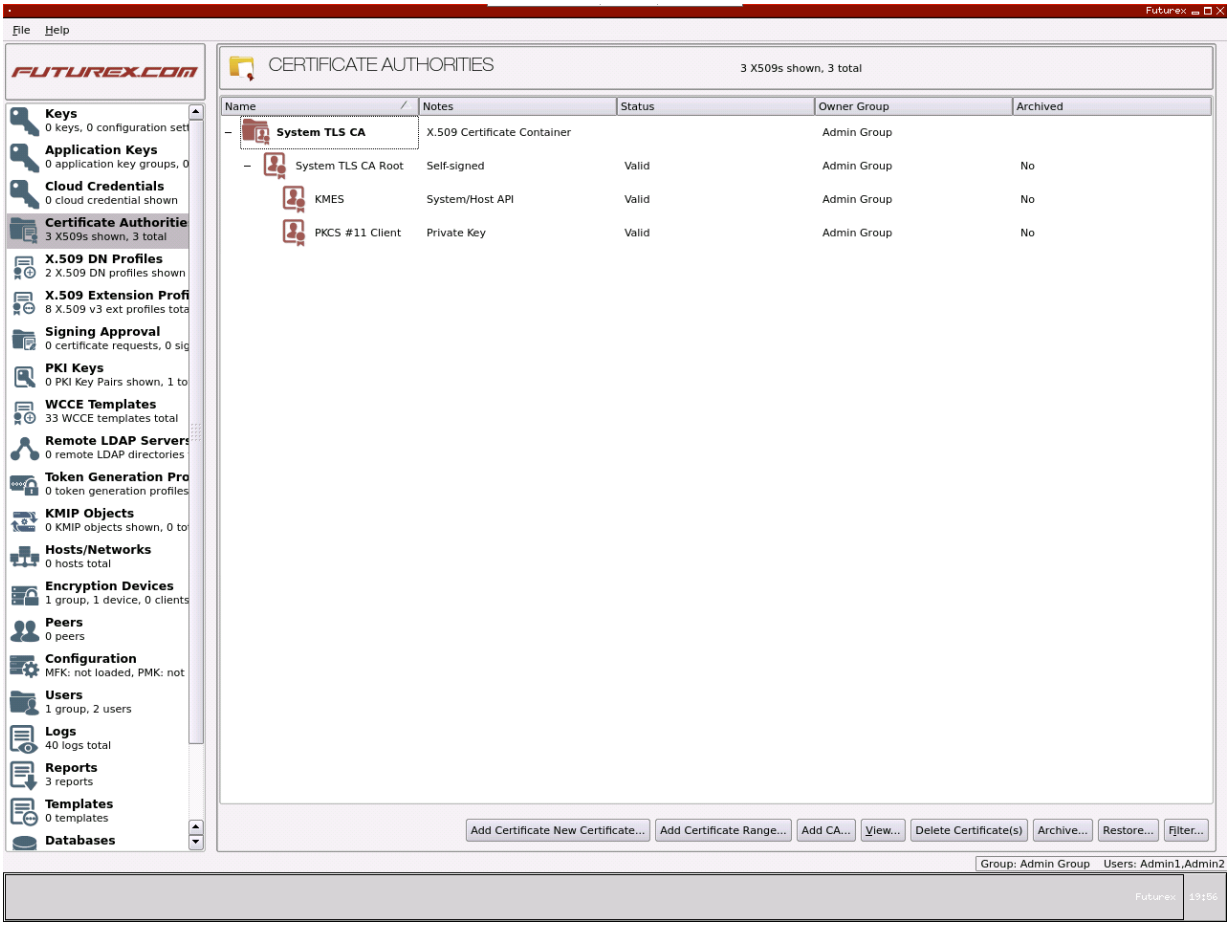


8. Click **OK** to save and exit the Network Options dialog.

[3.1.6] Generate a signed certificate for the SSH client PKCS #11 library

1. Go to the *Certificate Authorities* menu.
2. Right-click on the root CA certificate and select **Add Certificate** -> **New Certificate...**
3. In the *Subject DN* tab, set a Common Name for the certificate, such as "PKCS #11 Client".
4. In the *Basic Info* tab, change the Major key to the **PMK**. All other settings can be left as the default values.
5. In the *V3 Extensions* tab, select the "Example TLS Client Certificate" profile, then click **OK**.

6. The signed PKCS #11 client certificate will be listed now under the root CA certificate.



[3.1.7] Export the signed certificate for the SSH client PKCS #11 library

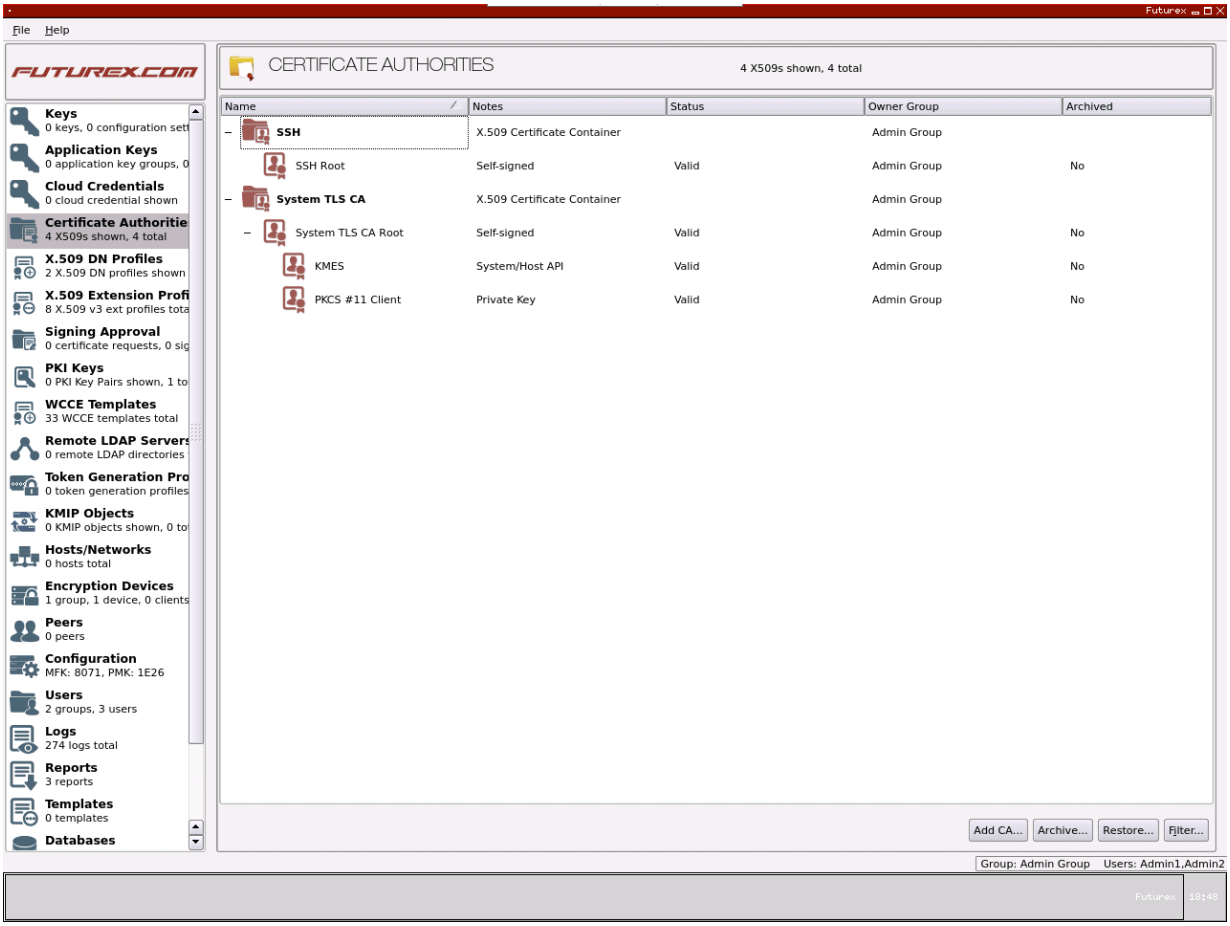
1. Go to the *Certificate Authorities* menu.
2. Right-click on the signed PKCS #11 client certificate, then select **Export** -> **PKCS12...**
3. In the *Export PKCS12* dialog, select **Export Selected** under Export Options, then click **Next >**.
4. Specify a password for the PKCS #12 file, then click **Next >**.
5. Click **Finish**.

NOTE: The `export_pkcs12.p12` file will be saved in the root directory of either the USB or SFTP mount point, depending on which is configured. This PKCS #12 file needs to moved to the SSH client, along with the root CA certificate. In a later section, the PKCS #11 library will be configured to use those certificates for TLS communication with the KMES Series 3.

[3.2] GENERAL KMES CONFIGURATIONS FOR SSH KEY OFFLOADING

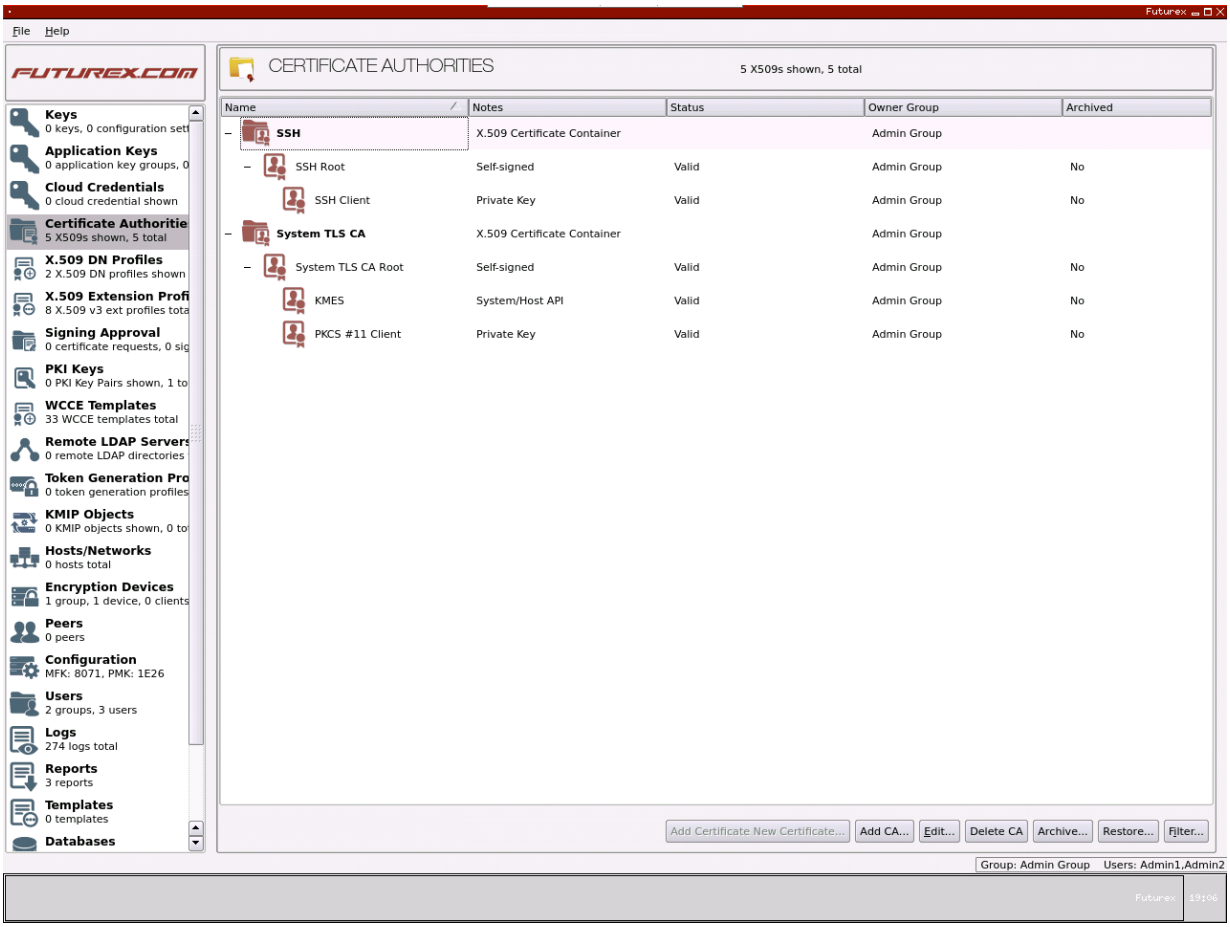
[3.2.1] Create a CA for SSH keys/certificates

1. Select *Certificate Authorities* in the left 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, such as "SSH". Leave all other fields as the default values, then click **OK**.
3. The Certificate Container that was just created will be listed now in the Certificate Authorities menu.
4. Right-click on the SSH Certificate Container and select **Add Certificate -> New Certificate...**
5. In the *Subject DN* tab, set a Common Name for the certificate, such as "SSH Root".
6. In the *Basic Info* tab, change the Major key to the **PMK**. All other settings can be left as the default values.
7. In the *V3 Extensions* tab, leave the Profile set to **None**, then click **OK**.
8. The root SSH certificate will be listed now under the SSH Certificate Container.



[3.2.2] Create the SSH client key/certificate

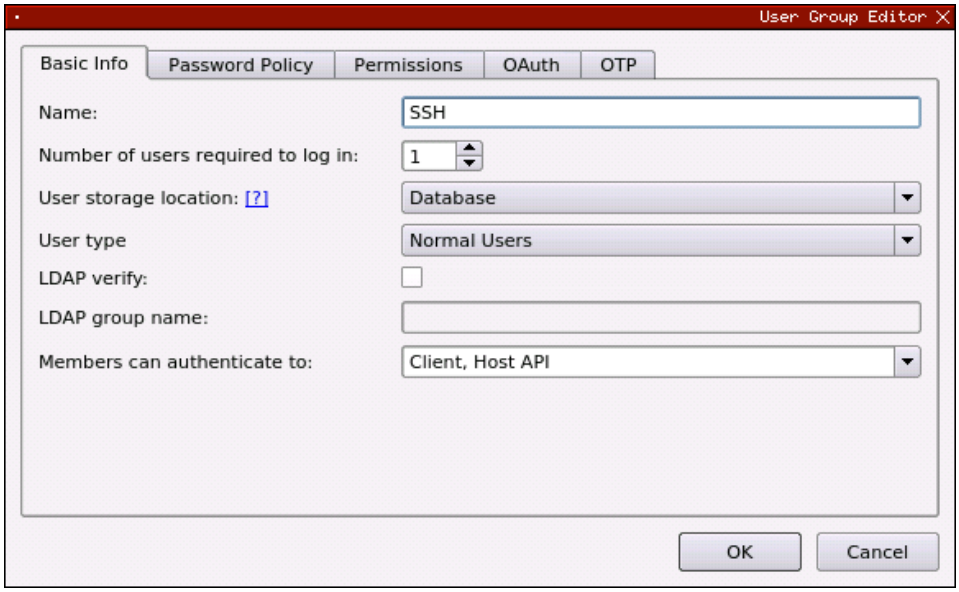
1. Right-click on the root SSH certificate and select **Add Certificate** -> **New Certificate...**
2. In the *Subject DN* tab, set a Common Name for the certificate, such as "SSH Client".
3. In the *Basic Info* tab, all settings can be left as the default values.
4. In the *V3 Extensions* tab, leave the Profile set to **None**, then click **OK**.
5. The signed SSH client certificate will be listed now under the root SSH certificate.



[3.2.3] Create an SSH User Group with the required permissions

1. Select *Users* in the left menu, then click the **Add Group...** button at the bottom of the page.

2. Specify a name for the group, such as "SSH", then ensure that the settings below are selected.

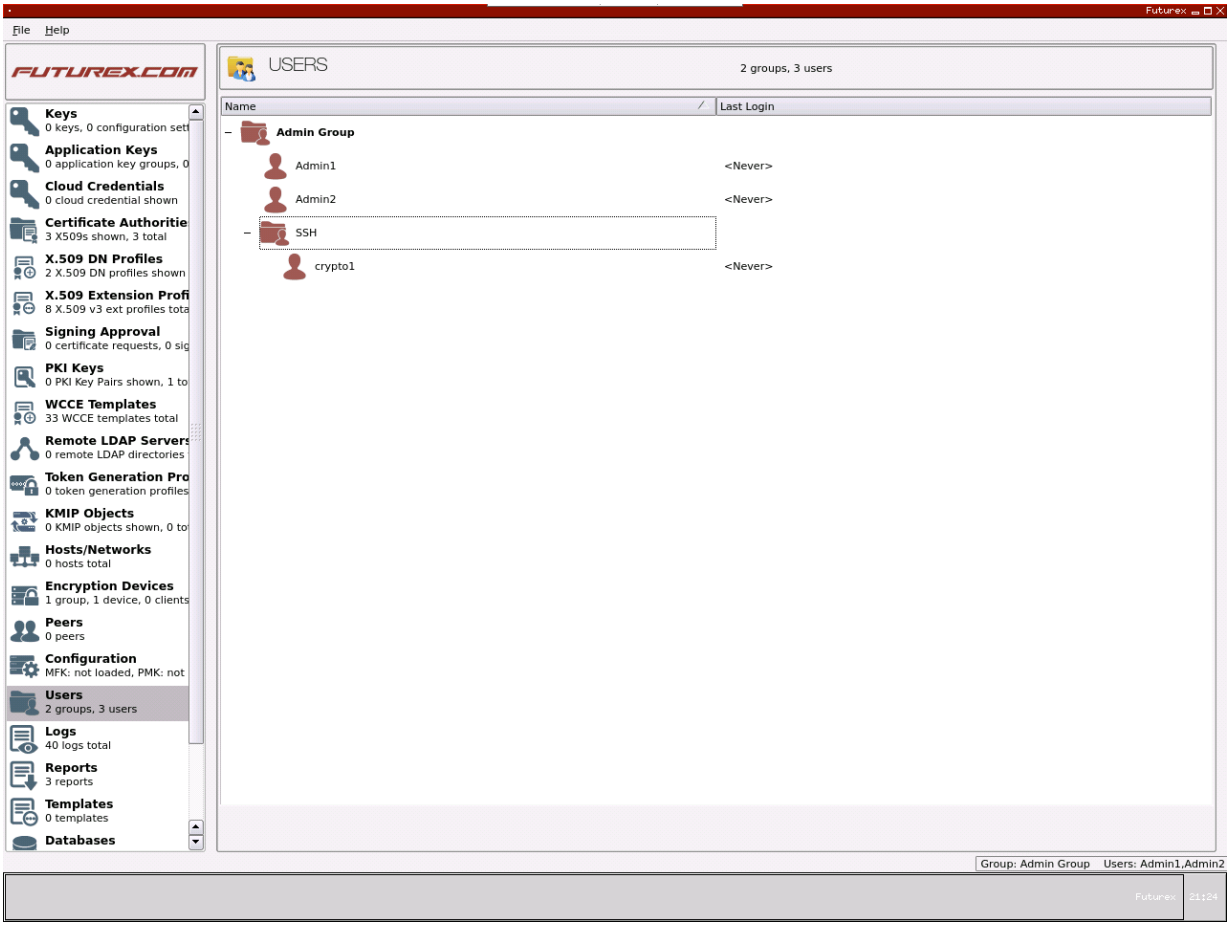


3. In the *Permissions* tab, ensure that only the following permissions are selected:
    - Manage certificates
    - Manage certificates
        - Export
        - Upload
    - Manage keys (only the top-level **Manage keys** permission)
    - Manage signing approvals
    - Manage signing approvals
        - Add
4. Click **OK** to save.
5. Disregard the warning that there are only 0 users currently in the group. Click **OK**.

[3.2.4] Create a single login user within the SSH User Group

1. On the *Users* page, right-click on the **SSH** User Group and select **Add** -> **User...**
2. Set a user name and password for the User. All of the other settings can be left as their default values.

3. Click **OK**, then you should see the new user listed under the **SSH** User Group, as shown below:



[3.2.5] Create a Key Group

1. Select *Keys* in the left menu, then click **Add Key Group...** at the bottom of the page.
2. In the *Key Group Editor* dialog, set a name for the Key Group and change the Owner group to **SSH**, then click **OK**.

[3.2.6] Create a Signing Approval Group and give the SSH User Group permissions to use it

1. Select *Signing Approval* in the left menu, then click the **Add Approval Group...** button at the bottom of the page.
2. Set a name for the Approval Group, such as "SSH", then click **OK** to save.
3. Right-click on the **SSH** Approval Group, then select **Permission...**
4. Give the **SSH** User Group the **Use** permission, then click **OK**.

[3.2.7] Grant the SSH User Group permissions to the SSH certificates

1. Go to the *Certificate Authorities* menu.
2. Right-click on the **SSH** Certificate Container, then select **Permission...**
3. Give the **SSH** User Group the **View** permission, then click **OK**.
4. Right-click on the **SSH Root** certificate, then select **Permission...**
5. Give the **SSH** User Group the **View** permission, then click **OK**.
6. Right-click on the **SSH Client** certificate, then select **Permission...**
7. Give the **SSH** User Group the **Use** permission, then click **OK**.

[3.2.8] Apply an Issuance Policy to the SSH Client certificate

1. Go to the *Certificate Authorities* menu.
2. Right-click on the **SSH Client** certificate, then select **Issuance Policy** -> **Add...**
3. In the *Basic Info* tab, set Approvals to **0**.
4. In the *X.509* tab, set the Default approval group to **SSH**.
5. In the *Object Signing* tab, select the **Allow object signing** checkbox.
6. Click **OK** to apply the Issuance Policy to the **SSH Client** certificate.

[3.2.9] Export the public key for the SSH Client certificate

1. Go to the *Certificate Authorities* menu.
2. Right-click on the **SSH Client** certificate, then select **Export** -> **Public Key(s)...**

NOTE: The **SSH Client** public key needs to be moved to the SSH server.

[3.2.10] Enable the required Host API commands

1. Go to *Configuration* -> *Host API Options*.
2. Enable every command, with the exception of command such as CLCR or RKRL, which have nested commands.
   NOTE: It is important that none of the commands which have nested commands are enabled due to a bug that exists at the time of writing.
3. Click **Save**.

[4] CONFIGURATION ON THE SSH SERVER AND CLIENT

[4.1] CONFIGURE THE SSH CLIENT PUBLIC KEY ON THE SSH SERVER AND DISABLE NON-KEY-BASED MODES OF AUTHENTICATION

1. Log in to the SSH server machine as the root user.
2. Open a terminal session and navigate to the location of the SSH client public key file that was exported from the KMES Series 3.
3. Run the following OpenSSL command to convert the public key that was exported from the KMES Series 3 from DER format to PEM format (this is required for the `ssh-keygen` command in step 4):
   ```
   openssl rsa -RSAPublicKey_in -inform DER -outform PEM -in SSH_Client_Public.pub -out SSH_Client_Public.pem
   ```
4. SSH requires a specific format for the public keys that are used within an SSH session. Run the following `ssh-keygen` command to convert the `SSH_Client_Public.pem` file that was output from the previous command to the required SSH public key format, then add it to the `~/.ssh/authorized_keys` file:
   ```
   ssh-keygen -f SSH_Client_Public.pem -i -m PKCS8 >> ~/.ssh/authorized_keys
   ```
5. Open the SSH daemon's configuration file.
   ```
   vim /etc/ssh/sshd_config
   ```
6. Inside the file, make sure that the following directive is set:
   ```
   PubkeyAuthentication yes
   ```
7. Optionally, the following directives can be set as well to make the SSH daemon only respond to SSH keys:
   ```
   PasswordAuthentication no
   ChallengeResponseAuthentication no
   ```
8. Save and close the file when finished. To implement these changes, the SSH service must be restarted. On Ubuntu or Debian machines, the following command can be issued:
   ```
   sudo service ssh restart
   ```

   On CentOS/Fedora machines, the daemon is called sshd:
   ```
   sudo service sshd restart
   ```

[4.2] CONFIGURE THE FUTUREX PKCS #11 (FXPKCS11) LIBRARY ON THE SSH CLIENT

1. Log in to the SSH client machine as the root user.
2. Open the FXPKCS11 configuration file (`fxpkcs11.cfg`) wherever that it is located on your machine with a text editor.

3. The **<KMS>** section should look similar to the following:

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

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

    # Key group name
    <KEYGROUP-NAME>         keygroup1               </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>           /futurex/root_cert.pem        </PROD-TLS-CA>
    <PROD-TLS-KEY>          /futurex/export_pkcs12.p12       </PROD-TLS-KEY>
    <PROD-TLS-KEY-PASS>     safest                  </PROD-TLS-KEY-PASS>

    <FX-LOAD-BALANCE>       NO                      </FX-LOAD-BALANCE>
</KMS>
```

Note the following:

- The values set in the **<CRYPTO-OPR>** and **<CRYPTO-OPR-PASS>** tags match what was set for the user that was created on the KMES Series 3.
- The value set in the **<KEYGROUP-NAME>** tag matches the name of the Key Group that was created on the KMES Series 3.
- The file set in the **<PROD-TLS-CA>** tag is the certificate that was exported from the root CA on the KMES Series 3.
- The file set in the **<PROD-TLS-KEY>** tag is the PKCS #12 file that was exported from the signed **PKCS #11 Client** certificate on the KMES Series 3.
- The value set in the **<PROD-TLS-KEY-PASS>** tag is the password for the PKCS #12 file defined in the **<PROD-TLS-KEY>** tag.

4. In the **<CONFIG>** section, the following define needs to be added so that FXPKCS11 will not prompt for a password during the SSH connection attempt:
   NOTE: It is not necessary for a password to be entered during the connection attempt because the FXPKCS11 library is already logged in to the KMES Series 3 with the configured user at that point.

```
<REQUIRE-LOGIN-FLAG>        NO                  </REQUIRE-LOGIN-FLAG>
```

[5] TEST AN SSH CONNECTION USING THE FXPKCS11 LIBRARY AND THE KMES SERIES 3

1. Log in to the SSH client machine as the root user.
2. Open a terminal session.
3. Run the following command to test an SSH connection to the SSH server using the FXPKCS11 library and the private key that is stored on the KMES Series 3. In the `-I` flag, the location of the FXPKCS11 library is specified.

```
ssh -I /usr/local/bin/fxpkcs11/libfxpkcs11.so root@SSH_Server_IP
```

4. If the above command is successful, the SSH client will be dropped into a remote SSH session on the SSH server without having to enter the password of the remote user that it's connecting with (i.e., root in this case).

```
root@SSH_Server_IP:~$
```

APPENDIX A: XCEPTIONAL SUPPORT

In today's high-paced environment, we know you are looking for timely and effective resolutions for your mission-critical needs. That is why our Xceptional Support Team will help do whatever it takes to ensure you have the best experience and support possible. Every time. Guaranteed.

- 24x7x365 mission critical support
- Level 1 to level 3 support
- Extremely knowledgeable subject matter experts

At Futurex, we strive to supply you with the latest data encryption innovations as well as our best-in-class support services. Our Xceptional Support Team goes above and beyond to meet your needs and provide you with exclusive services that cannot be found anywhere else in the industry.

- Technical Services
- Onsite Training
- Virtual Training
- Customized Consulting
- Customized Software Solutions
- Secure Key Generation, Printing, and Mailing
- Remote Key Injection
- Certificate Authority Services

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

ENGINEERING CAMPUS

864 Old Boerne Road

Bulverde, Texas, USA 78163

Phone: +1 830-980-9782

+1 830-438-8782

E-mail: info@futurex.com

XCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

SOLUTIONS ARCHITECT

E-mail: solutions@futurex.com