Connected
Key Management Server

Software Version 24.2.2

Installation and Upgrade Guide

opentext™

Document Release Date: May 2024
Software Release Date: May 2024

Legal notices

Acknowledgements

This product includes software written by Eric Young (eay@cyrptsoft.com).

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for updated documentation, visit https://www.microfocus.com/documentation/connected-mx/.

Support

Visit the MySupport portal to access contact information and details about the products, services, and support that OpenText offers.

This portal also provides customer self-solve capabilities. It gives you a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the MySupport portal to:

- View information about all services that Support offers
- Submit and track service requests
- Contact customer support
- Search for knowledge documents of interest
- View software vulnerability alerts
- Enter into discussions with other software customers
- Download software patches
- Manage software licenses, downloads, and support contracts

Many areas of the portal require you to sign in. If you need an account, you can create one when prompted to sign in.

Contents

Chapter 1: Overview

To keep your data secure while in motion as well as at rest, OpenText Connected uses escrowed encryption keys by default. Although these keys provide ample security for many customers, Connected also supports a customer-managed encryption key option for sites that require enhanced data security. When using this option, implemented by the Connected Key Management Server (Connected-KMS) component, you maintain full control over the encryption keys that protect your company's data. As a result, only those within your company ever have access to your valuable corporate data.

> **CAUTION:** Ensure that you follow best practices for key management by rotating and backing up your site-specific encryption keys. If your site's keys are compromised or lost, you lose access to your company's Connected data and Support cannot recover it for you .

Components

Managing your own encryption keys for Connected requires installation and configuration of the following components in your local environment:

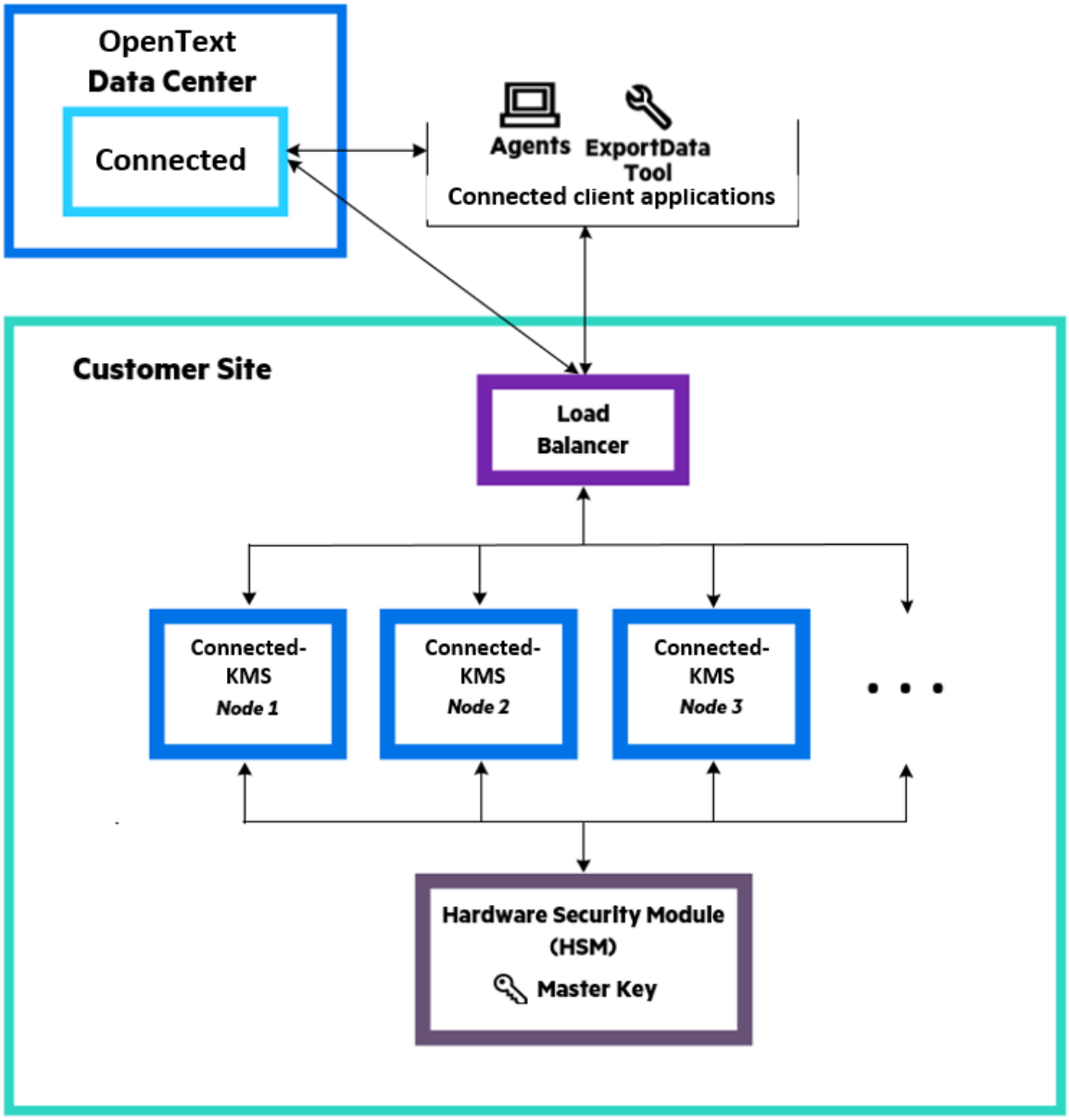
- **Connected Key Management Server (Connected-KMS)**. A Connected software server component that services Connected requests for encrypted and decrypted copies of your site's encryption keys. This component works in conjunction with your site's hardware security module (HSM) to fulfill these requests.
- **A physical or virtual hardware security module**. The HSM maintains your site's master encryption key. This device services encryption and decryption requests from Connected-KMS that require the use of your master encryption key. Neither Connected nor Connected-KMS ever have direct access to that key. One example of an HSM compatible with Connected-KMS is OpenText Enterprise Secure Key Manager (ESKM).
- **Load balancer (optional, but recommended)**. OpenText recommends that you deploy Connected-KMS in a distributed multi-node environment, serviced by a load balancer, such as HAProxy.

Deployment Model

Connected supports both single-node and multi-node deployments of the Connected-KMS component. However, OpenText recommends that you deploy Connected-KMS in a distributed multi-node environment, administered by a load balancer, such as HAProxy.

Using a multi-node environment improves the application's performance and reliability by distributing Connected requests across multiple nodes. Therefore, if a node goes offline or becomes inaccessible, other nodes remain available to service requests for your site's encryption keys. In contrast, if the node in a single-node deployment becomes inaccessible, Connected stops protecting user data and all data currently stored in it becomes inaccessible until access to the node is restored.

The following figure illustrates the major components in a multi-node Connected-KMS deployment and identifies the data flow between them.



This document provides the information necessary to install and configure Connected-KMS, including the configurations necessary for it to communicate with Connected, the user devices registered with it, and with the HSM. For HSM installation and configuration instructions, which are outside the scope of this document, see the HSM's documentation.

SSL Certificates

Connected-KMS uses several SSL certificates to establish secure connections with other components. There are two sets of Connected-KMS-related certificates: one for connecting with

Connected applications and another for connecting with the HSM's KMIP-server. Use the information in this section as a reference when performing the tasks in this document to create and install these certificates.

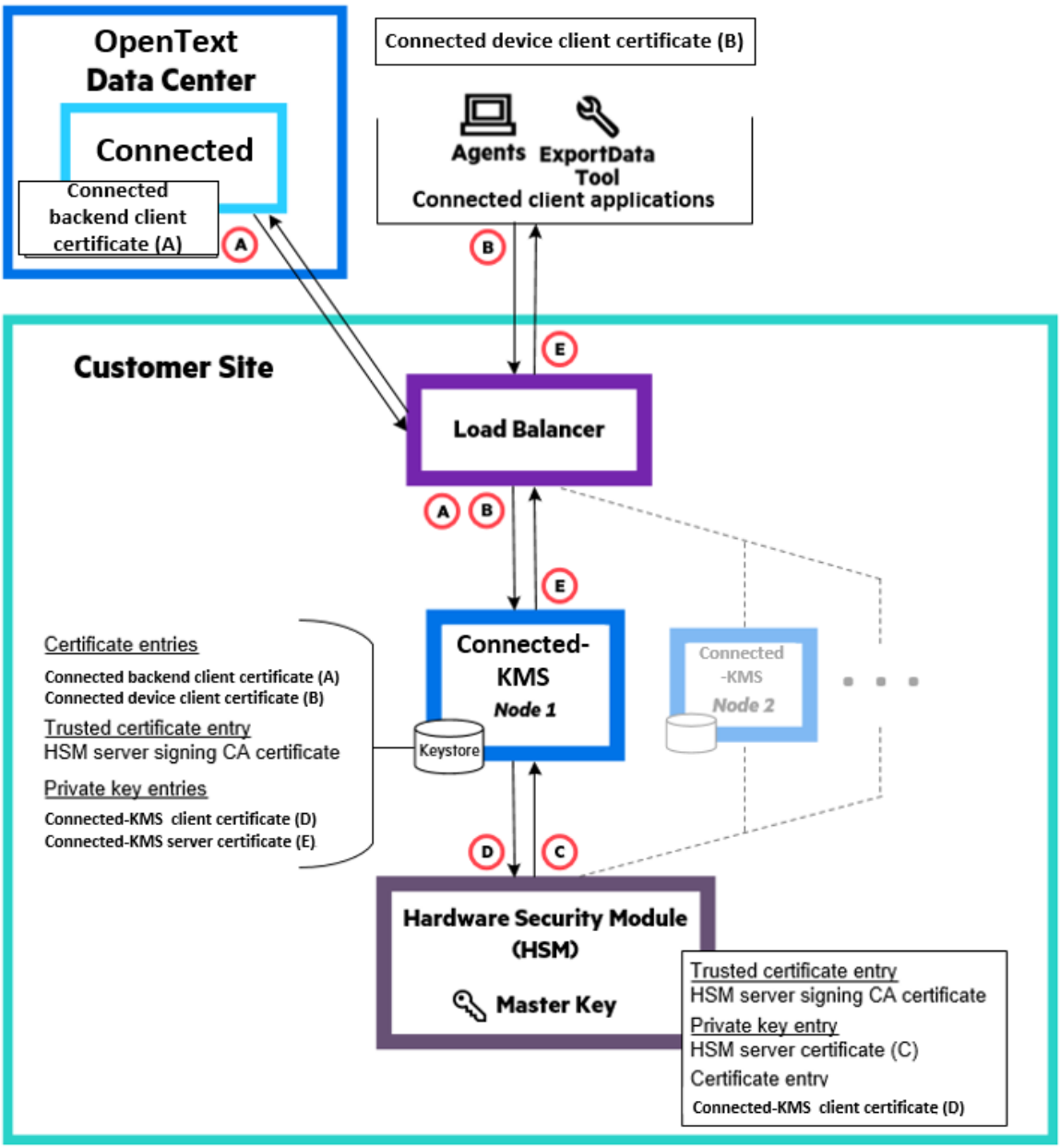
Certificates for connectivity with Connected applications:

- **Connected-KMS server certificate**. The certificate Connected-KMS uses to identify itself to clients, such as Connected applications.
- **Connected backend client certificate**. The certificate the Connected backend server uses to identify itself to Connected-KMS.
- **Connected device client certificate**. The certificate Connected client applications use to identify themselves to Connected-KMS. One certificate supports all Connected client applications in your environment.

Certificates for connectivity with the HSM's server:

- **Connected-KMS client certificate**. The certificate that Connected-KMS uses to identify itself to servers, such as the HSM's server.
- **HSM server certificate**. The certificate that the HSM's server uses to identify itself to clients. A valid KMIP server certificate is a prerequisite to installing Connected-KMS. For information about creating this certificate and installing it on your HSM, see your HSM's documentation.
- **HSM server signing CA certificate**. The certificate of the CA that signed the *HSM server certificate*. A valid KMIP server signing CA certificate is a prerequisite to installing Connected-KMS. For information about creating this certificate and installing it on your HSM, see your HSM's documentation.

The following figure shows the installed location of these certificates and the communication channels that use them.



Chapter 2: Install Connected-KMS

This chapter provides information about how to install and configure the Connected-KMS software.

Installation process

Before you begin

Ensure that your environment meets all hardware, software, and third-party component requirements as described in the *Connected-KMS Support Matrix*. You can access this guide from the Connected documentation site.

To install Connected-KMS

1. Download the Connected-KMS software package, `cmx-kms-server-rpm.tar.gz`, from the Connected web application to a temporary location in your environment accessible by all computers that will run Connected-KMS:
   a. Log in to the Connected web application using a Connected administrator or data administrator account.
   b. In the upper-right corner of the Connected web application, click your name, and then click **Downloads**.
   c. Under **Tools & Utilities** on the Downloads page, click **Download Key Management Server (Connected-KMS)**.

   During the installation process, you will copy this software to each Connected-KMS node in your environment.
2. If you plan to deploy Connected-KMS on multiple nodes, create a list that contains each node's host name and TCP port number on which it listens for incoming requests.

   You must specify these values while configuring each node, so it is best to create the list now.
3. Create the SSL certificates and keystore.
4. On each node in your Connected-KMS environment:
   a. Install the Connected-KMS software.
   b. Configure the Connected-KMS software.
5. In a multi-node environment, install and configure a load balancer, such as HAProxy, to distribute requests across all Connected-KMS nodes.

   Configure each node in the load balancer with port 8443, which is the port that each node listens to for requests. For detailed information about how to configure your load balancer, see the documentation that accompanies it.

6. Provide the following to OpenText Software Fulfillment or the OpenText Partner who will create your Connected corporate account:
   - PKSC12 file and password of the Connected backend client certificate
   - Password of the Connected device client certificate
   - Public URL of either the load balancer in a multi-node deployment or the Connected-KMS server in a single-node deployment
7. Install the Connected device client certificate.

Create the SSL certificates and Connected-KMS keystore

Connected-KMS uses SSL certificates to secure the channels it uses to communicate with other components. When establishing a communication channel, Connected-KMS sends its own certificate to identify itself and verifies the certificate it receives against those in its keystore. This chapter describes how to create these certificates and the Connected-KMS keystore.

Table 1 identifies the set of certificates that Connected-KMS uses, regardless of how many nodes you deploy, and provides notes about required signing and creation of each. For detailed information about these certificates and how the software uses them, see Overview, on page 4.

Table 1: Connected-KMS required certificates

Certificate	Notes
Connected backend client certificate	Signed by an external certificate authority (CA), such as VeriSign or DigiCert.
Connected device client certificate	Signed by an external CA.
Connected-KMS server certificate	Signed by an external CA. If you have an existing PKCS12 certificate for your Connected-KMS computer that meets this requirement, you can use it.
Connected-KMS client certificate	(**Optional** when using a Vault with Azure Authentication enabled) Signed by a CA that you have imported into the HSM as a Trusted Certificate Entry, such as the *HSM server signing CA certificate*. After creating this certificate using the steps in this document, you must import it into the HSM as a Certificate Entry. However, the steps to do so are outside the scope of this document. See the HSM documentation for details. If you have an existing PKCS12 certificate for your Connected-KMS computer that meets this requirement, you can use it.
HSM server signing CA certificate	You must create this certificate and import it into the HSM as a

Certificate	Notes
	Trusted Certificate Entry. To do so, see your HSM documentation for details. You must also import a copy of this certificate, in PEM format, into the Connected-KMS keystore file, as described in this chapter.
HSM server certificate	Signed by the *HSM server signing CA certificate*. You must create this certificate and import it into the HSM as a Private Key Entry. To do so, see the HSM documentation for details.

Create the certificates

Prior to Connected-KMS configuration, you must create the Connected backend client certificate, the Connected device client certificate, and the Connected-KMS server and client certificates. This section provides step-by-step instruction for creating these certificates using the OpenSSL toolkit and verifying them with Java Keytool. If your company uses different tools to create and verify certificates, you can use them.

Before you begin

Ensure that the HSM server certificate and HSM server signing CA certificate are available and installed on the HSM in the manner described in Table 1: Connected-KMS required certificates. For information about how to create these certificates and import them into your HSM, see the HSM documentation.

To create the SSL certificates

1. On one of your planned Connected-KMS nodes, create a `keystore` directory to hold all certificate files, and then go to the new directory.

   ```
   mkdir ~/keystore
   ```

   ```
   cd ~/keystore
   ```

2. Generate an RSA key.

   ```
   openssl genrsa -out keyName.key 2048
   ```

   This command stores an RSA key in the file *`keyName`*`.key`.

   Based on which certificate you want to create, specify *`keyName`* throughout these steps as follows or choose your own unique name to identify it. Examples throughout this document use the names listed.

   - **Connected backend client certificate**. `cmxBackend`
   - **Connected device client certificate**. `cmxDevice`

     Agents require this certificate to have the name `cmxDevice.p12`; therefore, by specifying the key name as `cmxDevice`, you do not have to rename the resulting certificate.

- **Connected-KMS server certificate**. `kmsServer`
- **Connected-KMS client certificate**. `kmsClient` (**Optional** for Vault)

> **NOTE:** The *`keyName`*`.key` file contains the RSA key in an unencrypted format, so keep this file secure.

3. Create a certificate signing request (CSR) for the RSA key.

   `openssl req -new -sha256 -key` *`keyName`*`.key -out` *`keyName`*`.csr`

   This command prompts for the following information:

   - **Country Name**. Two-letter Organization for Standardization (ISO) country code that identifies where your organization is legally registered. For example, type `US` for United States.
   - **State or Province Name**. Name of the state where your organization is legally located. Do not abbreviate this value. For example, type `Massachusetts` not `MA`.
   - **Locality Name**. Name of the city or town where your organization is legally located. Do not abbreviate this value. For example, type `New York City` not `NYC`.
   - **Organization Name**. Full legal name of your organization. Do not abbreviate this value. For example, type `Micro Focus` not `MF`.

     If the name of the Organization or Organizational Unit contains a symbol or special character (one that requires you to press the Shift key to type), either spell out or omit the symbol. For example, for AB&C Company, type `AB and C Company`.
   - **Organizational Unit Name**. (Optional) Name of the department or unit within your organization making the request. To keep this field blank, type a period (`.`).
   - **Common Name**. One of the following, depending on certificate:
     - **Connected backend client certificate**. A unique, case-sensitive, company-specific string in FQDN format that identifies the Connected backend server. Typically, `<`*`certIdentifier`*`>.<`*`company`*`>.com`

       For example, for the fictitious AcmeXYZ company: `cmxBackend.AcmeXYZ.com`.
     - **Connected device client certificate**. A unique, case-sensitive, company-specific string in FQDN format that identifies the Connected device. Typically, `<`*`certIdentifier`*`>.<`*`company`*`>.com`

       For example: `cmxDevice.AcmeXYZ.com`.
     - **Connected-KMS client certificate**. A unique, case-sensitive, company-specific string in FQDN format that identifies the Connected-KMS server as a client to the HSM. Typically, `<`*`certIdentifier`*`>.<`*`company`*`>.com`

       For example: `kmsClient.AcmeXYZ.com`.
     - **Connected-KMS server certificate**. The case-sensitive hostname (host + domain) of the Connected-KMS server. Typically, `<`*`server`*`>.<`*`company`*`>.com`

       For example: `myServer.AcmeXYZ.com`.

> **NOTE:** If prompted for a challenge password, optional company name, or email address, do not provide values. To leave those fields empty, type a period (`.`).

4. Submit the CSR to the appropriate CA, as described in Table 1: Connected-KMS required certificates.

   To proceed to the next step, you need the CA-signed certificate and possibly intermediate CA certificates in the trust chain so that you have all certificates up to, but not including, the Root CA. Usually the certificate trust chain contains an intermediate CA certificate; however, the use of none or more than one is possible. All certificates must be in PEM format, which typically have a `.crt` file extension. If the CA did not return an intermediate CA certificate, you can download it from the CA's website.

5. Create a certificate chain from your CA-signed certificate (*`keyName`*`.crt`), as follows:

   - If your CA-signed certificate is signed by the Root CA, your CA-signed certificate is essentially a chain that contains one certificate. Therefore, the remaining steps in this task refer to your *`keyName`*`.crt` file as the certificate chain named *`keynameCertChain`*`.crt`.
   - If your CA-signed certificate is signed by an intermediate CA, do the following:

     a. Create a copy of your CA-signed certificate in which to build the certificate chain.

        ```
        cp keyName.crt keynameCertChain.crt
        ```

     b. Open *`keynameCertChain`*`.crt` in a text editor, and then paste the entire contents of the intermediate CA certificate at the end of the file, starting on a new line.

        The file should now contain the following:

        ```
        -----BEGIN CERTIFICATE-----
        <Your CA-signed certificate in encoded format.>
        -----END CERTIFICATE-----
        -----BEGIN CERTIFICATE-----
        <Encoded certificate of the Intermediate CA that signed your
        certificate.>
        -----END CERTIFICATE-----
        ```

     c. If the trust chain for your certificate includes multiple intermediate CA certificates, add any remaining ones to the end of the file.

        First, add the intermediate CA certificate whose owner is the issuer of the intermediate CA certificate that you added in step b. Next, add any intermediate CA whose owner is the issuer of the previous certificate. Continue adding intermediate CA certificates in this order, ending with the one issued by a Root CA.

     d. Save the file.

6. If you added intermediate CA certificates to the *`keynameCertChain`*`.crt` file, verify its contents. To do so:

   a. List the certificate's contents.

      ```
      keytool –printcert -v -file keynameCertChain.crt
      ```

b. Verify that the certificates listed form a chain from your certificate, `Certificate[1]`, to one issued by a Root CA, `Certificate[N]`.

   `Certificate[1]` should identify your organization as the owner and an intermediate certificate authority as the issuer. For example, for the Connected backend client certificate:

   ```
   Certificate[1]:
   Owner: CN=cmxBackend, OU=MyOU, O=MyCorp, ST=MyState, C=US
   Issuer: CN=XYZ Intermediate CA, OU=www.xyzIntermediateCA.com, O=XYZOrg, C=US
   ```

   `Certificate[2]` should list its owner as the issuer of `Certificate[1]`. Its issuer can be another intermediate CA, or as in this example, a Root CA:

   ```
   Certificate[2]:
   Owner: CN=XYZ Intermediate CA, OU=www.xyzIntermediateCA.com, O=SomeOrg, C=US
   Issuer: CN=ABC Root CA, OU=www.abcRootCA.com, O=ABCOrg, C=US
   ```

   If the issuer is another intermediate CA, continue to follow the pattern matching each certificate's issuer to the following certificate's owner and ensure that the issuer of the last certificate is a Root CA.

   > **NOTE:** If the file does not contain the correct certificate chain, delete the file, and then repeat step 5 to create it again. To delete the file, type: `rm keynameCertChain.crt`

7. Create a password-protected PKCS12 file using the key generated in step 1 (*keyName*`.key`) and the file that contains your certificate chain (*keynameCertChain*`.crt`).

   ```
   openssl pkcs12 -export -inkey keyName.key -in keynameCertChain.crt -name
   "keyName" -out keyName.p12
   ```

   This command prompts for a password to encrypt the resulting PKCS12 file. The command does not display the password as you type it.

   > **NOTE:** Store this password in a secure location. You will need it and the PKCS12 file to create the keystore and to configure Tomcat's SSL properties.

8. Verify the PKCS12 certificate by examining its contents and ensuring it contains a certificate chain that includes your CA-signed certificate and intermediate CA certificates. To do so:

   a. List the certificate's contents.

      ```
      keytool -list -v -keystore keyName.p12 -storetype PKCS12 -storepass
      certPassword
      ```

      Where *certPassword* is the password that you assigned the PKCS12 file in the previous step.

   b. Verify that the following line exists, which indicates the certificate contains a private key:

      ```
      Entry type: PrivateKeyEntry
      ```

   c. Verify that the certificates listed form a chain from your certificate, `Certificate[1]`, to one ultimately issued by the Root CA, `Certificate[N]`.

Use the same logic as in step 6b to match each certificate's issuer to the following certificate's owner. The issuer of the last certificate must be a Root CA.

> **NOTE:** If the PKCS12 file does not contain the correct certificate chain, delete the file, and then repeat step 7 to create it again. To delete the file, type: `rm keyName.p12`

9. Repeat these steps until you have created all four certificates listed in step 1.

After you have created all certificates, you are ready to create the Connected-KMS keystore.

Create the Connected-KMS keystore

Connected-KMS uses a password-protected keystore to hold its copy of certificates that verify the identity of components requesting its services. This section provides step-by-step instruction for generating the Connected-KMS keystore using Java Keytool. If your company uses a different tool to create keystores, you can use it instead of Keytool in these steps.

Before you begin

Ensure that the Connected, Connected-KMS, and KMIP server signing CA certificates listed in Table 1: Connected-KMS required certificates, are available and that you know their passwords.

> **CAUTION:** The sample commands in this section include options for passwords. If you use the sample commands exactly as-is, the passwords that you assign will be stored in the computer's command history and therefore be viewable by others. If this poses a security issue in your environment, omit the password options from the commands. The Keytool command automatically prompts for any passwords not specified by the command.

To create the Connected-KMS keystore

1. (**Optional** when using the Vault) While still in the `keystore` directory, import the Connected-KMS client certificate into a new keystore file, *`kmsKeystore`*`.jks`.

   ```
   keytool -importkeystore -srckeystore kmsClient.p12 -srcstoretype PKCS12 \
   -srckeypass kmsClientPwd -srcstorepass kmsClientPwd -alias kmsClient \
   -deststorepass kmsKeystorePwd -destkeypass kmsClientKeyPwd \
   -destkeystore kmsKeystore.jks
   ```

   This command uses the following passwords:

   - ***`kmsClientPwd`***. Password that you gave the *`kmsClient`*`.p12` file when creating it.
   - ***`keystorePwd`***. Password to assign to the new keystore.
   - ***`kmsClientKeyPwd`***. Password to assign to private key of the keystore's *`kmsClient`* key pair. Connected-KMS requires that this password be the same as *`keystorePwd`*.

   > **TIP:** For more information about a specific option, type `keytool -option -help`.

2. Import the Connected-KMS server certificate into the keystore file.

   ```
   keytool -importkeystore -srckeystore kmsServer.p12 -srcstoretype PKCS12 \
   -srckeypass kmsServerPwd -srcstorepass kmsServerPwd -alias kmsServer \
   -deststorepass kmsKeystorePwd -destkeypass kmsServerKeyPwd \
   -destkeystore kmsKeystore.jks
   ```

   This command uses the following passwords:

   - ***kmsServerPwd***. Password that you gave the *kmsServer*.p12 file when creating it.
   - ***keystorePwd***. Password that you assigned to the new keystore in the previous step.
   - ***kmsServerKeyPwd***. Password to assign to the private key of the keystore's *kmsServer* key pair. Connected-KMS requires that this password be the same as *keystorePwd*.

3. Import the HSM server signing CA certificate into the keystore file.

   ```
   keytool -import -trustcacerts -alias kmipSigningCA -file kmipSigningCA.crt
   -keystore kmsKeystore.jks -storepass keystorePwd
   ```

   The tool prompts you to verify that the certificate looks correct. If so, type `yes`. Otherwise, type `no`, and then run the command again with the correct certificate file.

4. Import the Connected backend client certificate into the keystore file.

   ```
   keytool -import -alias cmxBackend -file cmxBackend.crt -keystore
   kmsKeystore.jks -storepass kmsKeystorePwd
   ```

5. Import the Connected device client certificate into the keystore file.

   ```
   keytool -import -alias cmxDevice -file cmxDevice.crt -keystore kmsKeystore.jks
   -storepass kmsKeystorePwd
   ```

6. Verify the keystore contents.

   ```
   keytool -v -list -keystore kmsKeystore.jks -storepass kmsKeystorePwd
   ```

   The Connected-KMS keystore should contain these entries:

   - (**Optional** when using Vault) Private key entry for the Connected-KMS client certificate
   - Private key entry for the Connected-KMS server certificate
   - Trusted certificate entry for the KMIP server signing CA certificate
   - Certificate entry for the Connected backend client certificate
   - Certificate entry for the Connected device client certificate

   The information for each private key entry should show the full certificate chain. The first part of the chain should contain information that you provided about your environment while creating the certificate and the end of the chain should contain information about the signing CA.

7. In a multi-node environment, copy the keystore to each Connected-KMS node.

   Use the secure copy command (`scp`), or other method, to transfer the keystore to each node. For example:

   ```
   scp kmsKeystore.jks jdoe@kms1.myCorp.com:
   ```

This command copies the `kmsKeystore.jks` file from the local directory to the jdoe user's home directory on the computer named `kms1.myCorp.com`.

After copying the keystore to each node, complete the installation of certificates on the HSM by importing the Connected-KMS client certificate into it.

Import the Connected-KMS client certificate into the HSM

> **IMPORTANT:** This section is **Optional** when using a Vault.

Connected-KMS presents its Connected-KMS client certificate to the HSM when trying to establish a secure connection. The HSM verifies the certificate against its local copy of the public key certificate before allowing the connection. You must install a copy of the Connected-KMS client certificate on the HSM for use in this verification process.

To import the Connected-KMS client certificate

- Import the CA-signed Connected-KMS client certificate (*`kmsClient`*`.crt`) into the HSM as a Certificate Entry (`CertEntry`).

  For information about how to import certificates into the HSM, see your HSM's documentation.

You are now ready to install the Connected-KMS software. To do so, see Install the Connected-KMS software, below.

Install the Connected-KMS software

You must install and configure the Connected-KMS software on each node in your Connected-KMS environment. The software, which you previously downloaded, comes packaged as a compressed TAR file that contains the following items:

- **Connected-KMS RPM (`cmx-kms-server-`*`version.build`*`.x86_64.rpm`)**. RPM package that installs the Connected-KMS software.
- **Installation script (`CMX-KMS-Installer.sh`)**. Installation script file that runs the Connected-KMS and Java SE RPMs.
- **Uninstallation script (`CMX-KMS-Uninstaller.sh`)**. Script that uninstalls the Connected-KMS software.

> **NOTE:** If Tomcat 7 is installed, it will be automatically uninstalled when installing Connected-KMS. It is recommended to copy the relevant data from /usr/share/tomcat.
>
> Tomcat 9 is automatically installed if not already installed.

To install the Connected-KMS software

1. On a computer you plan to install the Connected-KMS software, create a subdirectory under `/tmp` in which to copy and unzip the software package.

```
mkdir /tmp/connected-kms-installer
```

2. Go to the new directory.

   ```
   cd /tmp/connected-kms-installer
   ```

3. Copy the software package, `cmx-kms-server-rpm.tar.gz`, from where you previously downloaded it to the current directory, and then extract its contents.

   ```
   tar -xvf cmx-kms-server-rpm.tar.gz
   ```

4. Run the `CMX-KMS-Installer.sh` script with root privileges.

   ```
   sudo sh CMX-KMS-Installer.sh
   ```

   The script identifies the packages it will install and prompts for confirmation. Review this information and, if acceptable, enter `Y`.

   As the script runs, it provides information about the installation process. If an error occurs, correct the problem listed, and then run the script again.

   The Connected-KMS installation creates a `/usr/share/kms` directory for application files and a `/var/log/kms` directory for log files.

The software is now installed, but not configured. To configure it, see Configure the Connected-KMS software, below.

Configure the Connected-KMS software

Configure the Connected-KMS software that you previously installed on the computer.

To configure the Connected-KMS software

1. Copy the Connected-KMS keystore file, which you previously copied to your home directory, to the local `/usr/share/kms/conf/` directory.

   ```
   sudo cp ~/kmsKeystore.jks /usr/share/kms/conf
   ```

2. Configure the `kms.properties` file with site-specific values.

   a. Open the file `/usr/share/kms/conf/kms.properties` in a text editor.

      For example, to open the file using the vi editor, type:

      ```
      sudo vi /usr/share/kms/conf/kms.properties
      ```

      > **TIP:** vi is one of the most common editors available on Linux-based computers. You can find detailed information about how to use the vi editor on the Internet by searching for: `vi editor`.

   b. Configure the following `kms.properties` parameters:

      - **`kms.keystore.file=<keystorePath>`**. Path to the keystore file. For example: `/usr/share/kms/conf/kmsKeystore.jks`.

- **kms.keystore.password=<*keystorePassword*>**. Keystore password in plain text. The software also uses this password to access the Connected-KMS client certificate.
- (**Optional** when using a Vault) **kms.keystore.alias=<*keystoreAlias*>**. Case-insensitive alias that you provided when importing the Connected-KMS client certificate's PKCS12 file into the keystore. For example: `kmsClient`.
- **kms.hsm.server.url=<*kmipURL*>**. URL to the HSM's server. For example: `https://kmshsm.myCorp.com`

  > **NOTE:** The original property `kmip.server.url` is still supported but not preferred.

- **kms.hsm.server.port=<*kmipPort*>**. Port number on which the HSM's server listens for client requests, such as from Connected-KMS.

  > **NOTE:** The original property `kmip.server.port` is still supported but not preferred.

- The following additional properties to be configured **only** for the Vault:
  - **kms.hsm.protocol=<*Protocol*>**. The protocol used to communicate with the HSM. The default value is kmip1. For example, `kms.hsm.protocol=VAULT`
  - **kms.hsm.vault.namespace=<*Namespace*>**. The value to pass in the X-Vault-Namespace header. The default value is none. For example: `kms.hsm.vault.namespace=engineering`
  - **kms.hsm.vault.auth=<*token*>**. The authentication method to authenticate with the Vault. The default value is token. For example: `kms.hsm.vault.auth=azure`
  - **kms.hsm.vault.auth.renew-before-expiry=<*seconds*>**. Renew a token this number of seconds before the token expires. The default value is 30. For example: `kms.hsm.vault.auth.renew-before-expiry=35`
  - **kms.hsm.vault.auth.azure.role=<*role*>**. The role to pass during Azure Auth login. For example: `kms.hsm.vault.auth.azure.role=KMS_ADMIN`
  - **kms.hsm.vault.auth.azure.path=<*path*>**. The request path for the Azure Auth login request. The default value is `/v1/auth/azure/login`. For example: `kms.hsm.vault.auth.azure.path=/v1/auth/azure/login`
  - **kms.hsm.vault.auth.azure.resource=<*resource passed*>**. The resource passed in the Get Azure Token. The default value is `https://vault.hashicorp.com/`. For example: `kms.hsm.vault.auth.azure.resource=https://management.azure.com`

> **NOTE:** Keep the `kms.distributed.cache.config.file` parameter set to its default value.

c. In a mult-node environment, set `kms.cache.type=distributed`. Otherwise, keep the default value (none).

d. Save the file, and then exit the editor.

3. (**Only** for Vault) Configure the Vault so KMS can access it

a. Configure a Vault policy so KMS can use the Vault Transit Secrets Engine.

   Create a Vault policy that will be associated with the logged in KMS so it has the necessary permissions on the Transit Secrets engine. For example,

   ```
   #
   # Write a policy to allow KMS to create keys, get keys, and encrypt/decrypt
   (update). BTW,
   # as written this grants permission to all keys.
   #
   echo "Write <kms-app-policy>"
   vault policy write <kms-app-policy> - <<EOF
   path "transit/encrypt/*" {
   capabilities = [ "update"]
   }
   path "transit/decrypt/*" {
   capabilities = [ "update"]
   }
   path "transit/keys/*" {
   capabilities = [ "create",  "read", "update", "list" ]
   }
   EOF
   ```

   Where *kms-app-policy* is a name you give for the Vault policy.

b. Create a Vault Azure Auth role that refers to the policy created above.

   You configure this when running the Vault in Azure. For example,

   ```
   # Create a role. This role must also be specified in kms.properties.
   vault write auth/azure/role/kms \
   policies="kms-app" \
   bound_subscription_ids=<AZURE_SUBSCRIPTION_ID> \
   bound_resource_groups=<AZURE_RESOURCE_GROUP_NAME>
   ```

   For more information on Azure Authentication with HashiCorp Vault, refer the following resources:

   - https://www.hashicorp.com/resources/azure-authentication-hashicorp-vault
   - https://nedinthecloud.com/2019/01/23/using-azure-active-directory-authentication-with-hashicorp-vault-part-1/
   - https://github.com/straubt1/blogs.digitalinnovation/blob/master/blogs/vault-azure-and-terraform.md

4. Configure Tomcat with information about the keystore.

   a. Open the file `/usr/share/tomcat/conf/server.xml` in a text editor.

      For example, to open the file using the vi editor, type:

      ```
      sudo vi /usr/share/tomcat/conf/server.xml
      ```

   b. Update the keystoreFile, keystorePass and keyAlias appropriately in `server.xml`.

When doing so, set the following:

- **`keystoreFile`**. Path to the keystore file. For example: `/usr/share/kms/conf/kmsKeystore.jks`
- **`keystorePass`**. Keystore password in plain text. Tomcat also uses this password to access the Connected-KMS server certificate.
- (**Only** for KMIP) **`keyAlias`**. Case-insensitive alias that you provided when importing the Connected-KMS server certificate's PKCS12 file into the keystore. For example: `kmsServer`.

> **NOTE:** Ensure that port 8443 is publicly accessible so that Connected can establish communication with Connected-KMS through it.
> In a multi-node environment, port 8443 need not be publicly accessible, but the load balancer port needs to be publicly accessible.

c. Save the file, and then exit the editor.

5. In a multi-node environment, define the node in Tomcat's configuration file.

a. Open the file `/usr/share/tomcat/bin/setenv.sh` in a text editor.

For example, to open the file using the vi editor, type:

```
sudo vi /usr/share/tomcat/bin/setenv.sh
```

b. Locate the following comment line:

```
#JAVA_OPTS="${JAVA_OPTS} -Djgroups.tcp.address=<ipAddress>
-Djgroups.tcp.port=<portNum>"
```

Then, edit it as follows:

- Remove the leading number sign (#) to uncomment the line.
- Replace <*ipAddress*> with the host name of the node that you are installing. For example: `kms1.AcmeXYZ.com`
- Replace <*portNum*> with the TCP port number that the node uses to communicate with other Connected-KMS nodes in the cluster. To use the default port of 7800, omit the `-Djgroups.tcp.port` option.

> **NOTE:** You can designate a unique port for each node; however, the port must be accessible through firewalls.

The following example uses the default TCP port:

```
JAVA_OPTS="${JAVA_OPTS} -Djgroups.tcp.address=kms1.AcmeXYZ.com"
```

c. Save the file, and then exit the editor.

6. In a multi-node environment, define the set of all other Connected-KMS nodes in your distributed environment.

a. Open the file `/usr/share/kms/conf/jgroups.xml` in a text editor.

   For example, to open the file using the vi editor, type:

   ```
   sudo vi /usr/share/kms/conf/jgroups.xml
   ```

b. Locate the following section:

   ```
   <TCPPING
   initial_hosts=""
   />
   ```

   Then, specify the following:

   - **`initial_hosts`**. Comma-separated list of the hostname and TCP port number of each node in the cluster *except* the node that you are currently configuring. The syntax for this value is in the format: *`kmsHostName1`*`[`*`tcpPort1`*`],`*`kmsHostName2`*`[`*`tcpPort2`*`],`*`kmsHostName3`*`[`*`tcpPort3`*`]...`

   For example, the following line configures the first node of a four-node cluster in which all nodes listen for TCP requests on the default port 7800:

   ```
   <TCPPING
   initial_hosts="kms2.myCorp.com[7800],kms3.myCorp.com[7800],kms4.myCorp.com
   [7800]"
   />
   ```

c. Save the file, and then exit the editor.

7. Restart the Tomcat server.

   ```
   sudo systemctl restart tomcat
   ```

   > **TIP:** To configure the Tomcat server to start automatically each time the node restarts, use the following command:
   >
   > ```
   > sudo systemctl enable tomcat
   > ```

Connected-KMS software configuration on this node is complete. To install and configure the software on another node, see Install the Connected-KMS software, on page 16. Otherwise, complete the installation process by performing the remaining steps in Install Connected-KMS, on page 8.

Install the Connected device client certificate

Connected applications use the Connected device client certificate to establish a secure connection to Connected-KMS. Therefore, to ensure a successful connection, you must install this certificate on each device that has a Connected application installed.

> **NOTE:** Connected-KMS applications will not work correctly unless the Connected device client certificate is installed on the device.

Install the certificate for Agents

The Agent reads the Connected device client certificate from disk. You can install the certificate in either the user's or system's application data folder, depending on which location best meets your needs. For example, if multiple Connected users share the computer, install the certificate in the system's folder so that it applies to all users. You can use the instructions provided in this task to install the certificate manually or you can deploy it using a custom process that distributes and installs it on one or more clients at the same time.

> **NOTE:** Installation of the certificate file is outside of the Agent installation process. As a result, the file remains after you uninstall the Agent using the typical uninstall process. To remove the certificate after uninstalling the Agent, manually delete it.

Before you begin

Ensure that the name of the Connected device client certificate is `cmxDevice.p12`.

The Agent installation process creates the folder in which you will store the certificate. Therefore, install the Agent (but do not sign in) before installing the certificate. The Agent needs the certificate to encrypt and decrypt data, so sign in only after installing the certificate.

To install the certificate for Agents prior to Key Management Server

- Distribute the `cmxDevice.p12` certificate to the computer, and then place it into one of the following locations:
  - On Windows-based computers, either:
    - **system-specific location**. `%ALLUSERSPROFILE%\Connected`
    - **user-specific location**. `%LOCALAPPDATA%\Connected`
  - On macOS-based computers, either:
    - **system-specific location**. `/Library/Application Support/Connected`
    - **user-specific location**. `$HOME/Library/Application Support/Connected`

To install the certificate for Key Management Server Agents

- Distribute the `cmxDevice.p12` certificate to the computer, and then place it into one of the following locations:
  - On Windows-based computers, either:
    - **system-specific location**. `%ALLUSERSPROFILE%\Connected`
    - **user-specific location**. `%LOCALAPPDATA%\Connected`
  - On macOS-based computers, either:
    - **system-specific location**. `/Library/Application Support/Connected`
    - **user-specific location**. `$HOME/Library/Application Support/Connected`

Update cacert.pem

Update the cacert.pem if the KMS certificate is not signed by well known root CAs.

Windows

1. Under **user-specific location**. `%LOCALAPPDATA%\Connected` (for Key Management Server Agents) and `%LOCALAPPDATA%\Connected` (for prior to Key Management Server Agents), locate the cacert.pem.
2. Update cacert.pem with root CA which is used to sign KMS certificate.

macOS

1. Under **user-specific location**. `$HOME/Library/Application Support/Connected` (for Key Management Server Agents) and `$HOME/Library/Application Support/Connected` (for prior to Key Management Server Agents), locate the cacert.pem.
2. Update cacert.pem with root CA which is used to sign KMS certificate.

Convert the legacy .p12 or .pfx files to support OpenSSL 3

> **NOTE:** If you have later versions of Connected Agent 24.1, say 24.2 etc., and client certificates are created with older version of OpenSLL 3 tool kit, then convert the legacy .p12 or .pfx files to support OpenSSL3. OpenText recommends to follow the below steps to convert the files.

To Convert the legacy .p12 or .pfx files

1. Copy the temporary legacy certificate:

   ```
   cp cmxDevice.p12 temp_cmxDevice.p12
   ```

2. Convert the certificate in PEM format.

   ```
   openssl pkcs12 -legacy -in temp_cmxDevice.p12 -out temp_cmxDevice.pem -nodes -
   passin pass:CERTIFICATE_PASSWORD
   ```

   > **NOTE:** The CERTIFICATE_PASSWORD is the actual password of the certificate.

3. Extract the private Key

   ```
   openssl pkcs12 -legacy -in temp_cmxDevice.p12 -nocerts -out temp_cmxDevice.key
   -passin pass:CERTIFICATE_PASSWORD -passout pass:'CERTIFICATE_PASSWORD'
   ```

4. Creating new certicate:

   ```
   openssl pkcs12 -export -out new_cmxDevice.p12 -inkey temp_cmxDevice.key -in
   temp_cmxDevice.pem -passin pass:CERTIFICATE_PASSWORD -passout pass:CERTIFICATE_
   PASSWORD
   ```

5. Delete temporary files

   ```
   rm -rf temp_cmxDevice.p12 temp_cmxDevice.pem temp_cmxDevice.key cmxDevice.p12
   ```

6. Rename the new file name with

   ```
   mv new_cmxDevice.p12 cmxDevice.p12
   ```

Install the certificate for the ExportData tool

The ExportData tool reads the Connected device client certificate from disk and uses it to establish a secure connection with Connected-KMS. Although the certificate is password-protected, you can provide additional security by restricting access to the tool's root folder to only users who will run the tool.

Before you begin

Ensure that the name of the Connected device client certificate is `cmxDevice.p12`.

To install the certificate for the ExportData tool

- Copy the Connected client device certificate, `cmxDevice.p12`, to the ExportData tool's root folder.

> **NOTE:** Each instance of the ExportData tool requires its own certificate. Therefore, if you install the tool in multiple locations, you must install a copy of the certificate in the tool's root folder of each location.

Chapter 3: Upgrade Connected-KMS

This chapter provides information about how to upgrade the Connected-KMS software from a previous version.

You must upgrade the Connected-KMS software on each node in your Connected-KMS environment. The software, which you will download, comes packaged as a compressed TAR file that contains the following items:

- **Connected-KMS RPM (`cmx-kms-server-`*`version`*`.`*`build`*`.x86_64.rpm`)**. RPM package that installs the Connected-KMS software.
- **Installation script (`CMX-KMS-Installer.sh`)**. Installation script file that runs the Connected-KMS and Java SE RPMs.
- **Uninstallation script (`CMX-KMS-Uninstaller.sh`)**. Script that uninstalls the Connected-KMS software.

> **NOTE:** If Tomcat 7 is installed, it will be automatically uninstalled when installing Connected-KMS. It is recommended to copy the relevant data from /usr/share/tomcat.
>
> Tomcat 9 is automatically installed if not already installed.

Before you begin

Ensure that your environment meets all hardware, software, and third-party component requirements as described in the *Connected-KMS Support Matrix*. You can access this guide from the Connected documentation site.

To upgrade the Connected-KMS software

1. Download the Connected-KMS software package, `cmx-kms-server-rpm.tar.gz`, from the Connected web application to a temporary location in your environment accessible by all computers that will run Connected-KMS:
   a. Log in to the Connected web application using a Connected administrator or data administrator account.
   b. In the upper-right corner of the Connected web application, click your name, and then click **Downloads**.
   c. Under **Tools & Utilities** on the Downloads page, click **Download Key Management Server (Connected-KMS)**.

   During the upgrade process, you will copy this software to each Connected-KMS node in your environment.
2. Create a subdirectory under `/tmp` in which to copy and unzip the software package.

   ```
   mkdir /tmp/connected-kms-installer
   ```

3. Go to the new directory.

   ```
   cd /tmp/connected-kms-installer
   ```

4. Copy the software package, `cmx-kms-server-rpm.tar.gz`, from where you previously downloaded it to the current directory, and then extract its contents.

   ```
   tar -xvf cmx-kms-server-rpm.tar.gz
   ```

5. Make a copy of the existing Connected-KMS software so you can revert to it, if necessary.

   ```
   cp -r /usr/share/kms/conf /usr/share/kms/conf-orig
   ```

6. Run the `CMX-KMS-Installer.sh` script with root privileges.

   ```
   sudo sh CMX-KMS-Installer.sh
   ```

   The script identifies the packages it will upgrade and prompts for confirmation. Review this information and, if acceptable, enter `Y`.

   As the script runs, it provides information about the upgrade process. If an error occurs, correct the problem listed, and then run the script again.

   The Connected-KMS installation creates a `/usr/share/kms` directory for application files and a `/var/log/kms` directory for log files.

7. If replacing Oracle Java with an OpenJDK distribution, do the following:

   a. Stop tomcat:

      ```
      sudo systemctl stop tomcat
      ```

   b. Open the file `/usr/share/tomcat/bin/setenv.sh` in a text editor.

      For example, to open the file using the vi editor, type:

      ```
      sudo vi /usr/share/tomcat/bin/setenv.sh
      ```

   c. Set `JAVA_HOME` to the location where OpenJDK is installed.

      For example:

      ```
      JAVA_HOME="/usr/lib/jvm/jre"
      ```

   d. Save the file, and then exit the editor.

   e. Start tomcat:

      ```
      sudo systemctl start tomcat
      ```

8. Repeat steps 2 through 7 for each remaining node in your Connected-KMS environment.

Chapter 4: Replace certificates

To replace the certificates in your Connected-KMS environment, such as if they are expiring, use the following tasks:

- Replace Connected device client certificate

  This certificates identifies Connected client applications to Connected-KMS. One certificate supports all Connected client applications in your environment.

- Replace Connected backend client certificate

  This certificate identifies the Connected backend server to Connected-KMS.

- Replace Connected-KMS server certificate

  This certificate identifies Connected-KMS to Connected client applications.

Replace Connected device client certificate

Connected applications use the Connected device client certificate to establish a secure connection with Connected-KMS. This task provides the steps to replace this certificate, such as if it is expiring.

Before you begin

Make a backup copy of the following:

- `/usr/share/kms/conf` folder
- `cmxDevice.p12` certificate

To replace the Connected device client certificate

1. Obtain a new or renewed Connected device client certificate that has the *same* password as the current certificate.

   For information about how to create a new certificate, see Create the certificates, on page 10. Although the task provides information for creating all certificates used with Connected-KMS, you need only create the type that you want to replace, such as the Connected device client certificate. You will use the resulting `.p12` and `.crt` files in the steps that follow.

2. Import the Connected device client certificate (`.crt` file) into the keystore as follows:

   a. On one of your Connected-KMS nodes, navigate to the `keystore` directory that holds the new certificate file, and run the following command:

      `keytool -import -alias` *`cmxDeviceAlias`* `-file` *`cmxDevice`*`.crt -keystore` *`kmsKeystore`*`.jks -storepass` *`kmsKeystorePwd`*

To support multiple device certificates in the keystore, specify a unique *cmxDeviceAlias*. For example, append the current date, such as `cmxDevice20190512`. Using a unique alias enables clients to access Connected with their current certificate until you replace it. For Agents, this ensures backups are not interrupted.

b. In a multi-node environment, copy the keystore to each Connected-KMS node.

Use the secure copy command (`scp`), or other method, to transfer the keystore to each node. For example:

```
scp kmsKeystore.jks jdoe@kms1.myCorp.com:
```

This command copies the `kmsKeystore.jks` file from the local directory to the jdoe user's home directory on the computer named `kms1.myCorp.com`.

NOTE: Do not remove the current device certificate from the `kmsKeystore.jks` keystore file until after you have installed the new Connected device client certificate on all devices. This ensures that each device can continue to establish a secure connection with Connected-KMS using its current certificate until you replace it.

3. Replace the Connected device client certificate (`.p12` file) on each computer.

For information, see Install the Connected device client certificate, on page 21.

NOTE: Keep the following in mind when replacing the Connected device client certificate on a computer running an Agent:

- You can install the certificate in one of two locations. To ensure that Agents function correctly, locate and replace the current Connected device client certificate so that only one resides on the device.
- If, before you replace the certificate, the Agent icon in the notification tray indicates a certificate-related issue (such as if it expired), the error might not clear until the Agent attempts its next backup.

Replace Connected backend client certificate

This task provides the steps to replace the Connected backend client certificate, which the Connected backend server uses to identify itself to Connected-KMS.

Before you begin

Make a backup copy of the following:

- `/usr/share/kms/conf` folder
- *`cmxBackend`*`.p12` and *`cmxBackend`*`.crt` certificates

To replace the Connected backend client certificate

1. Obtain a new or renewed Connected backend client certificate.

   For information about how to create a new certificate, see Create the certificates, on page 10. Although the task provides information for creating all certificates used with Connected-KMS, you need only create the type that you want to replace, such as the Connected backend client certificate.

2. Import the Connected backend client certificate (`.crt` file) into the keystore.

   To do so, on one of your Connected-KMS nodes, navigate to the `keystore` directory that holds the new certificate file, run the following command, and then answer the password prompt:

   `keytool -import -alias cmxBackendAlias -file cmxBackend.crt -keystore kmsKeystore.jks`

   To avoid disruption of backups and restores, specify a unique *`cmxBackendAlias`*. For example, append the current date, such as `cmxBackend20190512`.

3. In a multi-node environment, copy the keystore to all other nodes as descried in Distribute updated keystore.

   > **TIP:** After successfully distributing the keystore to all other nodes in your environment, you can delete the original Connected backend client certificate.

4. In the Connected Web application, update your customer-managed key information as follows:
   a. Click the **HIERARCHY** tab.
   b. Select your company, and then click **EDIT**.
   c. Under **Additional Customer Features**, locate the **Customer-managed keys** option.
   d. To update certificate-related items, click **CONFIGURE CERTIFICATES**, and then do the following:
      i. To replace the Connected backend client certificate, click **Choose File**, and then use the Open dialog box to select the new Connected backend server certificate (`.p12`).

         If you select a file, and then decide that you want only to change a password and not to upload a new certificate, click **Clear**.

      ii. To replace a certificate's password, click **Edit** in the appropriate section, and then enter the new password.
   e. Click **SAVE**.

Replace Connected-KMS server certificate

This task provides the steps to replace the Connected-KMS server certificate, which Connected-KMS uses to identify itself to Connected client applications.

Before you begin

Make a backup copy of the following:

- *kmsServer*.p12 certificate

To replace the Connected-KMS server certificate

1. Obtain a new or renewed Connected-KMS server certificate.

   For information about how to create a new certificate, see Create the certificates, on page 10. Although the task provides information for creating all certificates used with Connected-KMS, you need only create the type that you want to replace, such as the Connected-KMS server certificate.

2. Import the Connected-KMS server certificate into the keystore.

   To do so, on one of your Connected-KMS nodes, navigate to the keystore directory, run the following command, answer the password prompts, and then confirm that the current certificate with kmsserver alias should be replaced:

   ```
   keytool -importkeystore -srckeystore kmsServer.p12 -srcstoretype PKCS12 \
   -srcAlias kmsServerAlias -destkeystore kmsKeystore.jks -destAlias kmsserver
   ```

   When specifying the command:

   - For srcAlias, set *kmsServerAlias* to the name you specified when converting the .crt file to a .p12 file.
   - For destAlias, always specify kmsserver because that alias is the value of the keyAlias parameter in the Tomcat server.xml file.

   When run, this command prompts for the following passwords:

   - ***kmsServerPwd***. Password that you gave the *kmsServer*.p12 file when creating it.
   - ***keystorePwd***. Password of the keystore.
   - ***kmsServerKeyPwd***. Password assigned to the private key of the keystore's *kmsServer* key pair. Specify the same password as for *keystorePwd*.

3. In a multi-node environment, copy the keystore to all other nodes as descried in Distribute updated keystore.

Distribute updated keystore

In a multi-node environment, after updating the keystore with a new certificate, copy the keystore to each Connected-KMS node.

To distribute the updated keystore

1. Copy the keystore to each Connected-KMS node.

   Use the secure copy command (`scp`), or other method, to transfer the keystore to each Connected-KMS node. For example:

   ```
   scp kmsKeystore.jks jdoe@kms1.myCorp.com:
   ```

   This command copies the `kmsKeystore.jks` file from the local directory to the jdoe user's home directory on the computer named `kms1.myCorp.com`.

2. On each Connected-KMS node, perform the following steps:

   a. Copy the keystore to the `/usr/share/kms/conf` directory.

   b. Restart tomcat using these commands:

   ```
   systemctl stop tomcat
   systemctl start tomcat
   ```

Chapter 5: Uninstall Connected-KMS

This chapter describes how to uninstall the Connected-KMS software from your environment. The uninstall process unlinks the `kms.war` file and removes the `/usr/share/kms` directory created during install. It does not delete the Connected-KMS software package or extracted install and uninstall scripts, if they still exist on the computer.

Before you begin

To uninstall Connected-KMS you need the `CMX-KMS-Uninstaller.sh` script that is part of the software package extracted onto each computer during Connected-KMS installation or upgrade. If this script is missing from a computer, you can copy it from another one. If it no longer exists on any computer in your environment, download the Connected-KMS software package and extract it again. To do so:

1. Download the Connected-KMS software package, `cmx-kms-server-rpm.tar.gz`, from the Connected web application to a temporary location in your environment accessible by all computers running Connected-KMS:

   a. Log in to the Connected web application using a Connected administrator or data administrator account.

   b. In the upper-right corner of the Connected web application, click your name, and then click **Downloads**.

   c. Under **Tools & Utilities** on the Downloads page, click **Download Key Management Server (Connected-KMS)**.

   The uninstall script is not version specific so you can use the one from the currently available version of Connected-KMS.

2. Go to the location where you downloaded the package and extract its contents.

   ```
   tar -xvf cmx-kms-server-rpm.tar.gz
   ```

To uninstall the Connected-KMS software

1. On a computer that contains Connected-KMS software you want to uninstall, go to the local directory that contains the `CMX-KMS-Uninstaller.sh` script.

   You might find the script in `/tmp/cmx-kms-installer`, which is the sample extraction directory the installation and upgrade documentation uses. However, if you cannot find it there or elsewhere on the computer, copy it from another computer to any temporary local directory.

2. Run the `CMX-KMS-Uninstaller.sh` script with root privileges.

   ```
   sudo sh CMX-KMS-Uninstaller.sh
   ```

   The script identifies the packages it will remove and prompts for confirmation. Review this information and, if acceptable, enter `Y`.

As it runs, the script provides information about the uninstall process. If an error occurs, correct the specified problem, and then run the script again.

> **NOTE:** The script might leave some files related to Connected-KMS in `/var/log/kms` and `/usr/share/kms`. If you no longer plan to use Connected-KMS, you may delete these directories.

3. Repeat these steps for each remaining node in your Connected-KMS environment.

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on OpenText Connected Key Management Server 24.2.2 Installation and Upgrade Guide

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to swpdl.ConnectedMx.DocFeedback@microfocus.com.

We appreciate your feedback!