Managing TLS Certificate, KeyStore, and TrustStore Files

This chapter contains the following sections:

About the TLS Certificate, KeyStore, and TrustStore Files

Note When Cisco Nexus Data Broker is started in a normal way, the connection to the device is HTTP. When Cisco Nexus Data Broker is started using the TLS protocol, the connection to the device is in HTTPS.

Enabling the TLS connections between Cisco Nexus Data Broker and the OpenFlow switches requires TLS KeyStore and TrustStore files. The TLS KeyStore and TLS TrustStore files are password protected.

Cisco Nexus Series switches connecting to Cisco Nexus Data Broker over OpenFlow require additional credentials, including Private Key, Certificate, and Certificate Authority (CA).

- The TLS KeyStore file contains the private key and certificate information used by Cisco Nexus Data Broker.
- The TLS TrustStore file contains the Certification Authority (CA) certificates used to sign the certificates on the connecting switches.

If TLS connections are required in your Cisco Nexus Data Broker implementation, all of the connections in the network must be TLS encrypted, and you must run Cisco Nexus Data Broker with TLS enabled. After Cisco Nexus Data Broker is started with TLS, you must run the TLS KeyStore password configuration command to provide the passwords for Cisco Nexus Data Broker to unlock the KeyStore files.

Preparing to Generate the TLS Credentials

OpenFlow switches require cryptographic configuration to enable TLS.

The NX-API protocol plugin now supports TLS for secure communication to the devices. You can connect to the NX-API protocol plugin on the secure port 443. All configuration, discovery, and statistics collection is done using secure communication. Cisco Nexus Data Broker should be configured with the required certificates and it should be started in the secure mode. When Cisco Nexus Data Broker is started in TLS mode, all devices support the TLS connection. The normal unencrypted connection to the switches is not accepted.

Caution Self-signed certificates are appropriate only for testing in small deployments. For additional security and more granular controls over individual certificate use and revocation, you should use certificates generated by your organization's Certificate Authority. In addition, you should never use the keys and certificates generated by this procedure in a production environment.

Before You Begin

Ensure that OpenSSL is installed on the Linux host where these steps will be performed.

SUMMARY STEPS

1. Create a TLS directory using **mkdir -p TLS** command and then navigate to it using **cd TLS** command:
2. Set up the directories for your CA system to function within. Create three directories under `mypersonalca` using **mkdir -p mypersonalca/<directory name>** command. To initialize the `serial` file and the `index.txt` file, enter **echo "01" > mypersonalca/serial** command and **touch mypersonalca/index.txt** command respectively.
3. Create the CA configuration file (ca.cnf). Before saving the ca.cnf file, some changes need to be made that are specific to the devices. One critical change is to change the [alt_names] section in the ca.cnf file to be relevant to the device IP address, because these IP addresses should be specified in the configuration file. If you need more or fewer IP/DNS names, you can add or remove the lines.
4. Once the directory structure is created and the configuration file (ca.cnf) is saved on your disk, create the TLS certificate file.
5. Copy **server.key** and **server.crt** into respective devices and install by using the following commands:
6. Creating the TLS KeyStore File
7. Creating the TLS TrustStore File
8. Starting application with TLS

DETAILED STEPS

Step 1 Create a TLS directory using **mkdir -p TLS** command and then navigate to it using **cd TLS** command:
mkdir -p TLS

cd TLS

Step 2 Set up the directories for your CA system to function within. Create three directories under `mypersonalca` using **mkdir -p mypersonalca/<directory name>** command. To initialize the `serial` file and the `index.txt` file, enter **echo "01" > mypersonalca/serial** command and **touch mypersonalca/index.txt** command respectively.
mkdir -p mypersonalca/certs

mkdir -p mypersonalca/private

mkdir -p mypersonalca/crl

echo "01" > mypersonalca/serial

touch mypersonalca/index.txt

The `serial` file and the `index.txt` file are used by the CA to maintain its database of the certificate files.

Step 3 Create the CA configuration file (ca.cnf). Before saving the ca.cnf file, some changes need to be made that are specific to the devices. One critical change is to change the [alt_names] section in the ca.cnf file to be relevant to the device IP address, because these IP addresses should be specified in the configuration file. If you need more or fewer IP/DNS names, you can add or remove the lines.

Note This step is applicable to NX-API only.

The following is an example of the content of the ca.cnf file:

```
[ ca ]
default_ca              = CA_default


[ CA_default ]
dir                     = .
serial                  = $dir/serial
database                = $dir/index.txt
new_certs_dir           = $dir/newcerts
certs                   = $dir/certs
certificate             = $certs/cacert.pem
private_key             = $dir/private/cakey.pem
default_days            = 365
default_md              = sha1
preserve                = no
email_in_dn             = no
nameopt                 = default_ca
certopt                 = default_ca
policy                  = policy_match
copy_extensions         = copy


[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional



[ req ]
default_bits            = 2048                  # Size of keys
default_keyfile         = example.key           # name of generated keys
default_md              = sha1                  # message digest algorithm
string_mask             = nombstr               # permitted characters
distinguished_name      = req_distinguished_name
req_extensions          = v3_req
x509_extensions         = v3_req
```

```
[ req_distinguished_name ]
# Variable name           Prompt string
#----------------------   ----------------------------------
0.organizationName      = Organization Name (company)
organizationalUnitName  = Organizational Unit Name (department, division)
emailAddress            = Email Address
emailAddress_max        = 40
localityName            = Locality Name (city, district)
stateOrProvinceName     = State or Province Name (full name)
countryName             = Country Name (2 letter code)
countryName_min         = 2
countryName_max         = 2
commonName              = Common Name (hostname, IP, or your name)
commonName_max          = 64


# Default values for the above, for consistency and less typing.
# Variable name                  Value
#------------------------------   ------------------------------
commonName_default             = www.cisco.com
0.organizationName_default     = Cisco
localityName_default           = San Jose
stateOrProvinceName_default    = CA
countryName_default            = US
emailAddress_default           = webmaster@cisco.com


[ v3_ca ]
basicConstraints        = CA:TRUE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer:always



[ v3_req ]
# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment


# Some CAs do not yet support subjectAltName in CSRs.
# Instead the additional names are form entries on web
# pages where one requests the certificate...
subjectAltName          = @alt_names


[alt_names]
IP.1   = 1.1.1.1
IP.2   = 2.2.2.2
IP.3   = 3.3.3.3
IP.4   = 4.4.4.4
```

```
[ server ]
# Make a cert with nsCertType set to "server"
basicConstraints=CA:FALSE
nsCertType                      = server
nsComment                       = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always


[ client ]
# Make a cert with nsCertType set to "client"
basicConstraints=CA:FALSE
nsCertType                      = client
nsComment                       = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
```

Step 4 Once the directory structure is created and the configuration file (ca.cnf) is saved on your disk, create the TLS certificate file.

Generate the TLS private key and Certification Authority (CA) files by entering the **openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out mypersonalca/certs/ca.pem -outform PEM -keyout mypersonalca/private/ca.key** command. This step generates the TLS private key in PEM format with a key length of 2048 bits and the CA file.

Generate the certificates (**server.key** and **server.crt**) file by entering **openssl req -new -x509 -days 365 -nodes -out server.crt -keyout server.key -config Example.conf**

Step 5 Copy **server.key** and **server.crt** into respective devices and install by using the following commands:

configure terminal to enter the configure terminal mode.

nxapi certificate httpskey keyfile bootflash:///server.key where bootflash:/// is a file location of **server.key**.

nxapi certificate httpscrt certfile bootflash:///server.crt where bootflash:/// is a file location of **server.crt**.

nxapi certificate enable

Step 6 Creating the TLS KeyStore File

Note The TLS KeyStore file should be placed in the configuration directory of Cisco Nexus Data Broker.

Copy **server.key** to **xnc-privatekey.pem**. This command copies the **server.key** file that was generated in step 5. For example, use the command **cp server.key xnc-privatekey.pem**.

Copy **server.crt** to **xnc-cert.pem**. This command makes a copy of the *server.crt* file that was generated in step 5. For example, use the command **cp server.crt xnc-cert.pem**.

Create the **xnc.pem** file, that contains the private key and certificate, by entering the **cat xnc-privatekey.pem xnc-cert.pem > xnc.pem** command.

Convert the PEM file **xnc.pem** file to the file **xnc.p12** file by entering the **openssl pkcs12 -export -out xnc.p12 -in xnc.pem** command. Enter a password at the prompt. This is the Export password. The password must contain at least 6 characters, for example, cisco123. You must use the same password for this step and for Step 7. The xnc.pem file is converted to a password-protected .p12 file.

Convert the xnc.p12 to a Java KeyStore (tlsKeyStore) file by entering the **keytool -importkeystore -srckeystore xnc.p12 -srcstoretype pkcs12 -destkeystore tlsKeyStore -deststoretype jks** command. This command converts the xnc.p12 file to a password-protected tlsKeyStore file. Enter a password at the prompt. Use the same password that you entered in previous step.

Step 7 Creating the TLS TrustStore File

The TLS TrustStore file should be placed in the application configuration directory.

Copy the **mypersonalca/certs/ca.pem** file to **sw-cacert.pem**.

Convert the **sw-cacert.pem** file to a Java TrustStore (tlsTrustStore) file by entering the **keytool -import -alias swca1 -file sw-cacert.pem -keystore tlsTrustStore** command.

Enter a password at the prompt. The **sw-cacert.pem** file is converted into a password-protected Java TrustStore (tlsTrustStore) file. The password must be at least six characters long, for example, *cisco123*

Step 8 Starting application with TLS

When Cisco Nexus Data Broker is running in TLS mode and if you restart Cisco Nexus Data Broker with the saved configuration, the **./runxnc.sh -start** command starts the application in TLS mode again. If you do not want to restart Cisco Nexus Data Broker in TLS mode, then delete the **configuration/startup/tlsconf.conf** file and start the application using the **./runxnc.sh -start** command again.

You do not need to provide the TLS KeyStore and TrustStore passwords when Cisco Nexus Data Broker is restarted with the saved configuration.