Managing TLS Certificate, KeyStore, and TrustStore Files

This chapter contains the following sections:

- About the TLS Certificate, KeyStore, and TrustStore Files, page 1
- Preparing to Generate the TLS Credentials, page 2
- Creating the TLS Private Key, Certificate, and Certification Authority, page 5
- Configuring the Cryptographic Keys on the Switch, page 5
- Enabling TLS for onePK and OpenFlow Switches, page 7
- Creating the TLS KeyStore File, page 8
- Creating the TLS TrustStore File, page 9
- Starting the Application with TLS Enabled , page 9
- Providing the TLS KeyStore and TrustStore Passwords, page 10

About the TLS Certificate, KeyStore, and TrustStore Files

Note To support onePK devices, all connections to Cisco Nexus Data Broker that use onePK or OpenFlow agents require Transport Layer Security (TLS).

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

Cisco Nexus 3000, 3100, and 3500 Series switches 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 (see Starting the Application with TLS Enabled , on page 9). After Cisco Nexus Data Broker is started with TLS, you must run the TLS KeyStore password configuration command (see Providing the TLS KeyStore and TrustStore Passwords, on page 10) to provide the passwords for Cisco Nexus Data Broker to unlock the KeyStore files.

Preparing to Generate the TLS Credentials

OpenFlow and Cisco onePK switches require cryptographic configuration to enable TLS.

Caution Self-signed certificates are appropriate only for testing in small deployments. For additional security, as well as 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.

Step 1 Create a TLS directory, and then navigate to it:
mkdir -p TLS

cd TLS

Step 2 Create three directories under `mypersonalca` and two prerequisite files:
mkdir -p mypersonalca/certs

mkdir -p mypersonalca/private

mkdir -p mypersonalca/crl

echo "01" > mypersonalca/serial

touch mypersonalca/index.txt

Step 3 Create the CA configuration file (ca.cnf).
The following is an example of the content of the ca.cnf file:

```
[ ca ]
default_ca = mypersonalca

[ mypersonalca ]
#
# WARNING: if you change that, change the default_keyfile in the [req] section below too
# Where everything is kept
dir = ./mypersonalca

# Where the issued certs are kept
```

```
certs = $dir/certs

# Where the issued crl are kept
crl_dir = $dir/crl

# database index file
database = $dir/index.txt

# default place for new certs
new_certs_dir = $dir/certs

#
# The CA certificate
certificate = $dir/certs/ca.pem

# The current serial number
serial = $dir/serial

# The current CRL
crl = $dir/crl/crl.pem

# WARNING: if you change that, change the default_keyfile in the [req] section below too
# The private key
private_key = $dir/private/ca.key

# private random number file
RANDFILE = $dir/private/.rand

# The extentions to add to the cert
x509_extensions = usr_cert

# how long to certify for
default_days = 365

# how long before next CRL
default_crl_days= 30

# which md to use; people in comments indicated to use sha1 here
default_md = sha1

# keep passed DN ordering
preserve = no

# Section names
policy = mypolicy
x509_extensions = certificate_extensions

[ mypolicy ]
# Use the supplied information
commonName = supplied
stateOrProvinceName = optional
countryName = optional
emailAddress = optional
organizationName = optional
```

```
organizationalUnitName = optional

[ certificate_extensions ]
# The signed certificate cannot be used as CA
basicConstraints = CA:false

[ req ]
# same as private_key
default_keyfile = ./mypersonalca/private/ca.key

# Which hash to use
default_md = sha1

# No prompts
prompt = no

# This is for CA
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
string_mask = utf8only
basicConstraints = CA:true
distinguished_name = root_ca_distinguished_name
x509_extensions = root_ca_extensions

[ root_ca_distinguished_name ]
# EDIT THOSE
commonName = Controller
stateOrProvinceName = Mass
countryName = US
emailAddress = root_ca_userid@cisco.com
organizationName = Cisco

[ root_ca_extensions ]
basicConstraints = CA:true
```

What to Do Next

Create the TLS certificate file.

Creating the TLS Private Key, Certificate, and Certification Authority

Before You Begin

Complete the steps in Preparing to Generate the TLS Credentials, on page 2.

Step 1 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.

Step 2 Generate the certificate key and certificate request files by entering the **openssl req -newkey rsa:2048 -keyout cert.key -keyform PEM -out cert.req -outform PEM** command.
This step generates the controller key (cert.key) and certificate request (cert.req) files in PEM format.

Important You must specify a PEM pass phrase that is 4 to 1024 alphanumeric characters in length, for example, cisco123.

You must also specify a common name in this step to complete Step 3. An example of a common name is the hostname of the server where Cisco Nexus Data Broker is running.

Step 3 Generate the certificate file by entering the **openssl ca -batch -notext -in cert.req -out cert.pem -config ca.cnf** command.
This step generates the certificate (cert.pem) file in PEM format using the certificate request (cert.req) and the certificate configuration (ca.cnf) files as inputs, and creates the certificates file (cert.pem) as output.
The following is an example of the console response:

```
Using configuration from ca.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'AU'
stateOrProvinceName   :ASN.1 12:'Some-State'
organizationName      :ASN.1 12:'Internet Widgits Pty Ltd'
commonName            :ASN.1 12:'localhost'
```

What to Do Next

Generate and import the certificate files on your Cisco Nexus 3000, 3100, or 3500 Series switches.

Configuring the Cryptographic Keys on the Switch

Before You Begin

Create the TLS certificate.

DETAILED STEPS

	Command or Action	Purpose
Step 1	switch(config)# **ip domain-name** *domain-name*	Configures the domain name for the switch.
Step 2	switch(config)# **crypto key generate rsa label myKey2 exportable modulus 2048**	Generates the cryptographic key.
Step 3	switch(config)# **crypto ca trustpoint myCA**	Enters the trustpoint configuration mode and installs the trustpoint file on the switch.
Step 4	switch(config-trustpoint)# **rsakeypair myKey2**	Installs the key files on the switch.
Step 5	switch(config-trustpoint)# **exit**	Exits trustpoint configuration mode.
Step 6	switch# **show crypto ca trustpoints**	(Optional) Verifies creation of the trustpoint files.
Step 7	switch# **show crypto key mypubkey rsa**	(Optional) Verifies creation of the key files.
Step 8	From the console, enter the **cat mypersonalca/certs/ca.pem** command.	Displays the certificate file on the machine hosting the generated TLS certificates.
Step 9	switch(config)# **crypto ca authenticate myCA**	Copies the CA certificate (`ca.pem`) to the switch to use as input. **Note** When copying the CA certificate, include the lines `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----`. End the input with a line that contains only `END OF INPUT`.
Step 10	switch(config)# **crypto ca enroll myCA**	Generates the certificate request on the switch.
Step 11	From the console, enter the **openssl ca -in n3k-cert.req -out newcert.pem -config ./ca.cnf** command.	Copies the certificate request from the switch to the file `n3k-cert.req` on your Linux machine, and then uses it to generate the switch certificate.
Step 12	switch(config)# **crypto ca import myCA certificate**	Copies the certificate (`newcert.pem`) to the switch.
Step 13	From the console, enter the **cat newcert.pem** command.	Displays the certificate on the Linux console.
Step 14	switch# **show crypto ca certificates**	Displays the certificates on the switch.

What to Do Next

Enable TLS for Cisco onePK and OpenFlow switches.

Enabling TLS for onePK and OpenFlow Switches

Before You Begin

- Create the TLS certificate.
- Configure the cryptographic keys on the switch.

DETAILED STEPS

	Command or Action	Purpose
Step 1	switch(config)# **onep**	Enters onePK configuration mode on the switch.
Step 2	switch(config-onep)# **transport type tls**	Enables TLS for onePK switches.
Step 3	switch# **exit**	Exits onePK configuration mode.
Step 4	switch# **show onep status**	(Optional) Displays the onePK configuration.
Step 5	switch(config)# **openflow**	Enters OpenFlow agent configuration mode on the switch.
Step 6	switch(config-ofa)# **switch 1**	Enters OpenFlow agent configuration mode for switch 1.
Step 7	switch(config-ofa)# **tls trust-point local myCA remote myCA**	Enables TLS certificate authority on the switch.
Step 8	switch(config-ofa-switch)# **pipeline** {*201*/*203*}	Configures the pipeline. **Note** Set the pipeline to **201** for Cisco Nexus 3000 and 3100 Series switches. This is the default value, and only expert users should set the number to any value other than 201. Set the pipeline to **203** for Cisco Nexus 3500 Series switches. This is the default value, and only expert users should set the number to any value other than 203.
Step 9	switch(config-ofa-switch)# **controller ipv4** {*A.B.C.D*} **port 6653 vrf management security tls**	Enables TLS for OpenFlow switches. *A.B.C.D* is the IP address of the controller. **Note** For more information about configuring TLS for OpenFlow (Cisco Nexus 3000, 3100, or 3500 Series switches), see the configuration guide for the switches in your environment.

What to Do Next

Create the TLS KeyStore file.

Creating the TLS KeyStore File

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

Before You Begin

Complete the steps in Configuring the Cryptographic Keys on the Switch.

Step 1 Copy `cert.key` to `xnc-privatekey.pem`.
This command copies the `cert.key` file that was generated in the "Creating the TLS Private Key, Certificate, and Certificate Authority" section. This file contains the Cisco Nexus Data Broker private key.

Step 2 Copy `cert.pem` to `xnc-cert.pem`.
This command makes a copy of the `cert.pem` file that was generated in the "Creating the TLS Private Key, Certificate, and Certificate Authority" section. This file contains the Cisco Nexus Data Broker certificate.

Step 3 Create the `xnc.pem` file, which contains the private key and certificate, by entering the **cat xnc-privatekey.pem xnc-cert.pem > xnc.pem** command.

Step 4 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.

Step 5 Enter a password at the prompt.

Note This is the Export password. Use the same password that you entered in Step 2 of "Creating the TLS Private Key, Certificate, and Certification Authority". 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.

Step 6 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

Step 7 Enter a password at the prompt.

Note Use the same password that you entered in Step 5.

Creating the TLS TrustStore File

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

Step 1 Copy the `mypersonalca/certs/ca.pem` file to `sw-cacert.pem`.

Step 2 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.

Step 3 Enter a password at the prompt.
The `sw-cacert.pem` file is converted into a password-protected Java TrustStore (tlsTrustStore) file.

Note The password must be at least six characters long, for example, **cisco123**.

Step 4 If the switches in your network use more than one CA certificate, repeat Step 1 through Step 3 for each CA certificate required.

Starting the Application with TLS Enabled

Before You Begin

- Generate and import certificate files on the switches.
- Enable TLS on the OpenFlow or onePK switches.
- Create and deploy TLS KeyStore and TLS TrustStore files for the Cisco Nexus Data Broker application.
- Make sure that the TLS KeyStore (tlsKeyStore) and TLS TrustStore (tlsTrustStore) files are located in the `./configuration` directory.

Step 1 From the console, start Cisco Nexus Data Broker by entering the **./runxnc.sh -tls -tlskeystore ./configuration/tlsKeyStore -tlstruststore ./configuration/tlsTrustStore** command.

Note You will not see any network elements until you provide the TLS KeyStore and TrustStore passwords as described in the next section.

Step 2 Cisco Nexus Data Broker is started with TLS enabled.

Providing the TLS KeyStore and TrustStore Passwords

The TLS KeyStore and TrustStore passwords are sent to the Cisco Nexus Data Broker so that it can read the password-protected TLS KeyStore and TrustStore files.

Step 1 Open a command window where you installed Cisco Nexus Data Broker.

Step 2 Navigate to the `xnc/bin` directory.

Step 3 Provide the TLS KeyStore and TLS TrustStore passwords by entering the **./xnc config-keystore-passwords** [**--user** {*user*} **--password** {*password*} **--url** {*url*} **--verbose --prompt --keystore-password** {*keystore_password*} **--truststore-password** {*truststore_password*}] command.
Enter the following information:

- The Cisco Nexus Data Broker username {*user*}—The user name.
- The Cisco Nexus Data Broker password {*password*}—The password for the user. For example, the default admin password is admin.
- The Cisco Nexus Data Broker web URL {*url*}—The web URL of the application. For example, the default URL is `https://Nexus_Data_Broker_IP:8443` or `http://Nexus_Data_Broker_IP:8080`.

  Note Use caution when using HTTP, because this will send your password to the controller in clear text.

- The TLS KeyStore password {*keystore_password*}—The TLS KeyStore password.
- The TLS TrustStore password {*truststore_password*}—The TLS TrustStore password.