

Managing TLS Certificate, KeyStore and TrustStore Files

This chapter has information and procedures for generating TLS Certificates and TrustStore files.

Beginning with Release 3.10.1, Cisco Nexus Data Broker (NDB) has been renamed to Cisco Nexus Dashboard Data Broker. However, some instances of NDB are present in this document, to correspond with the GUI, and installation folder structure. References of NDB/ Nexus Data Broker/ Nexus Dashboard Data Broker can be used interchangeably.

- Generating TLS Self-Signed Certification Between NDB Server and NDB Switch for NXAPI, on page 1
- Generating TLS 3rd Party Certification Between NDB Server and NDB Switch for NXAPI, on page 7
- Generating TLS Self-Signed Certification Between WebUI Browser and NDB Server, on page 14
- Generating TLS 3rd Party Certification Between WebUI Browser and NDB Server, on page 22

Generating TLS Self-Signed Certification Between NDB Server and NDB Switch for NXAPI

This section describes how to generate TLS self-signed certification between NDB server and NDB Switch. You need to generate certificates and keys for each switch to enable TLS. TLS communication between NDB switch and NDB server uses port 443 only.

Complete the following steps to generate TLS self-signed certification between NDB Server and NDB Switch for NXAPI:

- Generating Self-Signed Certificate and Key, on page 2
- Creating the TLS TrustStore File, on page 4
- Starting NDB with TLS, on page 5
- Configuring TLS KeyStore and TrustStore Passwords on Nexus Dashboard Data Broker , on page 6

Note You cannot configure a controller to communicate using port 80 after configuring TLS.

Generating Self-Signed Certificate and Key

This section describes how to generate self-signed certificate and key.

Before you begin

Ensure that you have domain name configured on the switch using **ip domain-name** command for each NDB switch that acts as the Fully Qualified Domain Name (FQDN) for the switch. For example:

```
conf t
ip domain-name cisco.com
hostname N9k-117
end
```

The FQDN for the switch is configured to N9K-117.cisco.com.

Procedure

Step 1 Log in to the server.

Step 2 Generate the private key and self-signed certificate using the **openssl req** command.

Example:

```
docker@docker-virtual-machine:~/TLS$ openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out
sw1-ca.pem -outform PEM -keyout sw1-ca.key

Generating a 2048 bit RSA private key
...+++
.........................................+++
writing new private key to 'sw1-ca.key'
-----
You are about to be asked to enter information that will be incorporated into
your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:SJ
Organization Name (eg, company) [Internet Widgits Pty Ltd]:cisco
Organizational Unit Name (eg, section) []:insbu
Common Name (e.g. server FQDN or YOUR name) []:N9K-117.cisco.com
Email Address []:myname@cisco.com
```

Note
If you have multiple switches, generate the certificate file and private key for each switch.

This command creates a certificate file (sw1-ca.pem) and a private key (sw1-ca.key).

Step 3 Log in to the NDB switch.

Step 4 Copy the certificate file, sw1-ca.pem, and keyfile, sw1-ca.key, to the switch using the **copy** command.

Example:

```
N9K-117# copy scp://docker@10.16.206.250/home/docker/Mallik/TLS_CA_june_23/sw1-ca.pem bootflash:
Enter vrf (If no input, current vrf 'default' is considered): management
docker@10.16.206.250's password:
server.cer
                                                                      100% 4676
   4.6KB/s   00:00
Copy complete, now saving to disk (please wait)...

N9K-117# copy scp://docker@10.16.206.250/home/docker/Mallik/TLS_CA_june_23/sw1-ca.key bootflash:
Enter vrf (If no input, current vrf 'default' is considered): management

docker@10.16.206.250's password:
cert.key
                                                                    100%
Copy complete, now saving to disk (please wait)...
```

Note
If you have multiple switches, repeat this step for all the switches.

Step 5 Configure the certificate file, sw1-ca.pem, and keyfile, sw1-ca.key in the switch using the **nxapi** command.

Example:

```
N9K-117 (config)# nxapi certificate httpskey keyfile bootflash:sw1-ca.key
Upload done. Please enable. Note cert and key must match.
N9K-117 (config)#
N9K-117 (config)# nxapi certificate httpscrt certfile bootflash:sw1-ca.pem
Upload done. Please enable. Note cert and key must match.
 N9K-117 (config)#
```

Note
If you have multiple switches, configure the corresponding certificate and private key to each switch.

Step 6 Enable self-signed certificates on the switch using the **nxapi certificate** command.

Example:

```
N9K-117 (config)# nxapi certificate enable
N9K-117 (config)#
```

Note
Ensure that there is no error while enabling self-signed certificates on the switch.

Step 7 Log in to the server.

Step 8 Copy and convert the sw1-ca.key and sw1-ca.pem files to .PEM format using the **copy** command.

Example:

```
cp sw1-ca.key sw1-ndb-privatekey.pem
cp sw1-ca.pem sw1-ndb-cert.pem
```

Step 9 Concatenate the private key and the certificate file using **cat** command.

Example:

```
docker@docker-virtual-machine:~/TLS$ cat sw1-ndb-privatekey.pem sw1-ndb-cert.pem > sw1-ndb.pem
```

Step 10 Convert the .pem file to .p12 file format using the **openssl** command. Enter the export password when prompted to create a password protected .p12 certificate file.

Example:

```
docker@docker-virtual-machine:~/TLS$openssl pkcs12 -export -out sw1-ndb.p12 -in sw1-ndb.pem
Enter Export Password: cisco123
Verifying - Enter Export Password: cisco123
Enter a password at the prompt. Use the same password that you entered in the previous Step (cisco123)
```

Step 11 Convert the sw1-ndb.p12 to a password protected Java KeyStore (tlsKeyStore) file using the **keytool** command. Use the jre/bin from the installed java directory.

Example:

```
docker@docker-virtual-machine:~/TLS$ ./(relativePath)/keytool -importkeystore -srckeystore sw1-ndb.p12
 -srcstoretype pkcs12 -destkeystore tlsKeyStore -deststoretype jks
Enter Destination Keystore password:cisco123
Re-enter new password:cisco123
Enter source keystore password:cisco123
Entry for alias 1 successfully imported.
Import command completed: 1 enteries successfully imported, 0 enteries failed or cancelled.
```

Note
By default an alias named "1" is stored in tlsKeyStore for the first switch. If the NDB controller is managing multiple switches, repeat this step for all the switches.When you add the second switch, the utility allows you to rename the first switch alias and also provides a provision to rename alias for the second switch. Refer examples as shown below.

```
keytool -importkeystore -srckeystore sw2-ndb.p12 -srcstoretype pkcs12 -destkeystore tlsKeyStore
-deststoretype jks
keytool -importkeystore -srckeystore sw3-ndb.p12 -srcstoretype pkcs12 -destkeystore tlsKeyStore
-deststoretype jks
```

Step 12 List and verify content in the java tlsKeyStore using the **keytool** command.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -list -v -keystore tlsKeyStore | more
```

What to do next

Proceed to the subsequent task, *Creating the TLS TrustStore File*.

Creating the TLS TrustStore File

TrustStore is created from the self-signed certificates that are generated for one or more switches. It holds certificates for one or more switches in the controller. This section describes how to create a Truststore using the self-signed certificate created in Generating self-signed certificate and key section. If you have multiple switches in the controller, each switch will have separate certificate file (For example, sw1-ndb-cert.pem, sw2-ndb-cert.pem)

Procedure

Step 1 Log in to the server.

Step 2 Convert the certificate file (For example, sw1-ndb-cert.pem) to a Java TrustStore (tlsTrustStore) file using the **keytool** command. Enter a password when prompted to create a password protected Java TrustStore (tlsTrustStore) file. The password should be at least six characters. Use the jre/bin installed in the java directory.

Example:

```
docker@docker-virtual-machine:~/TLS$ ./(relativePath)/keytool -import -alias sw1 -file sw1-ndb-cert.pem
 -keystore tlsTrustStore -storetype jks
Enter Export Password: cisco123
Verifying - Enter Export Password: cisco123
Enter a password at the prompt. Use the same password that you entered in the previous Step (cisco123)
```

Note
If a NDB controller manages multiple switches, repeat this step for all the switches to add all switch keys into the same TrustStore. For example:

```
docker@docker-virtual-machine:~/TLS$ keytool -import -alias sw2 -file sw2-ndb-cert.pem -keystore
tlsTrustStore
docker@docker-virtual-machine:~/TLS$ keytool -import -alias sw3 -file sw3-ndb-cert.pem -keystore
tlsTrustStore
// Here sw2 and sw3 are alias for switch 2 and switch 3 for identification purpose.
```

Step 3 List and verify keys for multiple switches in the same tlsTrustStore using the **keytool** command.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -list -v -keystore tlsTrustStore | more
```

Starting NDB with TLS

To start NDB with TLS, complete these steps:

Procedure

Step 1 Log in to the NDB server.

Step 2 Stop the NDB application, if running, using the **runndb.sh** command

Example:

```
 ./runndb.sh -stop
Controller with PID: 17426 -- Stopped!
```

Note
When onboarding a device, ensure to provide the FQDN or IP address of the device, that was provided during the certificate generation for that device.

Step 3 Copy the tlsKeystore and tlsTruststore files that you created to configuration folder of NDB (ndb/configuration).

Example:

```
cp tlskeystore /root/ndb/configuration
cp tlsTrustStore /root/ndb/configuration
```

Step 4 Start the NDB application with TLS using the **runndb.sh** script.

Example:

```
./runndb.sh -tls -tlskeystore ./configuration/tlsKeyStore -tlstruststore ./configuration/tlsTrustStore
```

Example:

To start NDB with default username (admin) and a non-default password (for example, pwd123):

```
./runndb.sh -osgiPasswordSync -tls -tlskeystore ./configuration/tlsKeyStore -tlstruststore
./configuration/tlsTrustStore
If ndb password is changed, OSGi webconsole password needs to be changed.
To set non-default OSGi webconsole password, enter ndb Admin Password [default]:
 (Type the non-default password which was set)
```

Note
To disable TLS, run the **./runndb.sh -notls** command. To disable TLS and start NDB, run the **./runndb.sh -notls -start** command. Before disabling TLS, ensure to *stop* NDB. After TLS is disabled, the port number for the devices connected to the NDB server should be changed to 80.

Configuring TLS KeyStore and TrustStore Passwords on Nexus Dashboard Data Broker

You need to configure TLS KeyStore and TrustStore passwords to enable Nexus Dashboard Data Broker to read password protected TLS KeyStore and TrustStore files. To configure TLS KeyStore and TrustStore passwords on Nexus Dashboard Data Broker , complete these steps:

Procedure

Step 1 Log in to the Nexus Dashboard Data Broker server.

Step 2 Navigate to bin directory.

Example:

```
cd ndb/bin
```

Step 3 Configure the TLS KeyStore and TrustStore passwords using the **ndb config-keystore-passwords** command.

Example:

```
./ndb config-keystore-passwords --user admin --password admin --url https://ip-address_localhost:8443
 --verbose --prompt --keystore-password keystore_password --truststore-password truststore_password
Please enter your password: <enter the NDB GUI admin password>
```

In case Nexus Dashboard Data Broker is configured with AAA (Tacacs/LDAP/Radius), and if the above command, **ndb config-keystore-passwords** fails, and you see a *401 unauthorized* error, then:

a. Go to `ndb` or `xnc` directory.

b. Stop the Nexus Dashboard Data Broker server using `./runndb.sh -stop`.

c. Enable the flag `enable.LocalUser.Authentication` by changing the value from *false* to *true* in the Nexus Dashboard Data Broker **config.ini** file.

d. Start the Nexus Dashboard Data Broker server using `./runndb.sh -start`.

e. Run the **ndb config-keystore-passwords** command again.

Note
In a HA environment, you need to run the above procedure for all the Nexus Dashboard Data Broker servers in the cluster.

After the TLS is enabled on Nexus Dashboard Data Broker , all the connections between Nexus Dashboard Data Broker server and Nexus Dashboard Data Broker switch are established using port 443. Ensure that you change device connections in Nexus Dashboard Data Broker to use port 443.

Up on successfully completing these steps, you can add nexus switch in the controller using port 443. Use FQDN of the switch to add the device to the Nexus Dashboard Data Broker controller.

You can verify the Certificate information using the WebUI Sandbox of the switch.

Generating TLS 3rd Party Certification Between NDB Server and NDB Switch for NXAPI

This section describes how to generate TLS 3rd party certification between NDB server and NDB Switch. You need to request for a separate certificate and key for each switch in you network. TLS communication between NDB switch and NDB server uses port 443 only.

Complete the following steps to generate TLS 3rd party certification between NDB Server and NDB Switch for NXAPI:

- Obtaining Certificates from a Certification Authority
- Generating the Java tlsKeyStore and tlsTrustStore for NDB Controller
- Starting NDB with TLS
- Configuring TLS KeyStore and Truststore Passwords on NDB

Note Complete all the steps under both the sections to ensure successful communication between the controller and the switch over TLS.

Obtaining Certificates from a Certification Authority

You can obtain certificate from a Certification Authority (CA) in two ways. You can either directly approach a CA for both the private key and certificate. The CA will generate a private key on your behalf along with the certificate that contains the public key with issuing CA's signature.

In the other approach, you can generate a private key using tools such as openssl and generate a Certificate Signing Request (CSR) to a certificate issuing authority. The CA generates the certificates with public key using the user identity information from CSR.

Before you begin

Ensure that you have domain name configured in the switch using **ip domain-name** command for each NDB switch that acts as the Fully Qualified Domain Name (FQDN) for the switch. For example:

```
conf t
ip domain-name cisco.com
hostname N9k-117
end
```

The FQDN for the switch is configured to N9K-117.cisco.com.

Procedure

Step 1 Log in to the server.

Step 2 Generate the private key (cert.key) and certificate signing request (cert.req) using openssl command.

Note
If you have multiple switches, generate the certificate file and private key for each switch.

Example:

```
docker@docker-virtual-machine:~/Mallik/TLS_CA$ openssl req -newkey rsa:2048 -sha256 -keyout cert.key
 -keyform PEM -out cert.req -outform PEM

Generating a 2048 bit RSA private key
...............+++
.....................+++
writing new private key to 'cert.key'
Enter PEM pass phrase:                          □ cisco123
Verifying - Enter PEM pass phrase:     □ cisco123
-----
You are about to be asked to enter information that will be incorporated into your certificate
request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:CA
Locality Name (eg, city) [Newbury]:SJ
Organization Name (eg, company) [My Company Ltd]:cisco
Organizational Unit Name (eg, section) []:insbu
Common Name (eg, your name or your server's hostname) []:N9K-117.cisco.com
Email Address []:myname@cisco.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: □ cisco123
An optional company name []:□ cisco123


docker@docker-virtual-machine: # ls
cert.key  cert.req
```

Step 3 Verify the CSR using the openssl command.

Example:

```
docker@docker-virtual-machine:~/Mallik/TLS_CA$ openssl req -noout -text -in cert.req
```

Step 4 The private key is generated with a security passphrase. You may need to unencrypt the private key. To remove the passphrase from the private key, use the openssl command.

Example:

```
docker@docker-virtual-machine:~/Mk/TLS_CA$ ls
cert.key  cert.req
docker@docker-virtual-machine:~/Mk/TLS_CA$cp cert.key  cert.keybkp
docker@docker-virtual-machine:~/Mk/TLS_CA$ rm cert.key
docker@docker-virtual-machine:~/Mk/TLS_CA$ openssl rsa -in cert.keybkp -out cert.key

Enter pass phrase for cert.keybkp: cisco123
```

Note
Repeat this step to remove passphrase from private keys for all the switches.

Note
Depending on the tier of the CA you choose, you can get up to three certificates (certificate chain) for each CSR. This means you get three certificates (root, intermediate and domain) from CA for each NDB switch. You need to check with CA to identify each type of certificate. Certificate naming convention might be different for different certifying authorities. For example: test-root-ca-2048.cer (root), test-ssl-ca.cer (intermediate), N9K-117.cisco.com.cer (domain).

Certificates are mostly shared in .PEM file format.

The cert.req file data needs to be submitted to 3rd party certification authority. Follow the relevant procedures and get the three (certificate) files.

Step 5 Create a single certificate file from the three certificate files using the **cat** command. The concatenation should be done in the following order, domain certificate, root certificate, and intermediate certificate. Syntax for **cat** command: *cat domain certificateroot certificateintermediate certificate > server.cer*.

Example:

```
$cat N9K-117.cisco.com.cer test-root-ca-2048.cer test-ssl-ca.cer > server.cer
```

Step 6 Edit the newly created server.cer file to separate the concatenated END and BEGIN lines. Do not delete anything in the file.

Example:

```
-----END CERTIFICATE----------BEGIN CERTIFICATE-----


////// Modify the above line like this by adding a line feed between the two.
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
```

Note
Repeat this step all the switches.

Step 7 Log into the NDB switch.

Step 8 Copy the private key (cert.key) and the certificate from CA (server.cer) to the switch using the copy command.

Example:

```
N9K-117# copy scp://docker@10.16.206.250/home/docker/Mallik/TLS_CA_june_23/server.cer bootflash:
Enter vrf (If no input, current vrf 'default' is considered): management
docker@10.16.206.250's password:
server.cer
                                                100% 4676     4.6KB/s   00:00
Copy complete, now saving to disk (please wait)...

N9K-117# copy scp://docker@10.16.206.250/home/docker/Mallik/TLS_CA_june_23/cert.key bootflash:
Enter vrf (If no input, current vrf 'default' is considered): management
docker@10.16.206.250's password:
cert.key
                                             100%
Copy complete, now saving to disk (please wait)...
```

Note
Repeat this step for all the switches.

Step 9 Configure the certificate file, sw1-ca.pem, and keyfile, sw1-ca.key in the switch using the **nxapi** command.

Example:

```
N9K-117 (config)# nxapi certificate httpskey keyfile bootflash:cert.key
Upload done. Please enable. Note cert and key must match.
N9K-117 (config)#
N9K-117 (config)# nxapi certificate httpscrt certfile bootflash:server.cer
Upload done. Please enable. Note cert and key must match.
 N9K-117 (config)#
```

Note
If you have multiple switches, configure the corresponding certificate and private key to each switch.

Step 10 Enable self-signed certificates on the switch using the **nxapi certificate** command.

Example:

```
N9K-117 (config)# nxapi certificate enable
N9K-117 (config)#
```

Note
Ensure that there is no error while enabling self-signed certificates on the switch.

Creating TLS Keystore and Truststore Files for NDDB Controller

NDDB uses certificates and keys to secure communication between switches. It stores the keys and certificates in keystores. These files are stored as tlsTruststore and tlsKeystore files in NDDB. Complete the following steps to generate the Java tlsKeyStore and tlsTrustStore files for NDDB Controller:

Procedure

Step 1 Copy the cert.key and server.cer created in the *Obtaining Certificates from a Certification Authority* section into the current directory (TLS). Select the certificate and key files for a single switch. These files were earlier generated for all the switches connecting to the controller. Using the server.cer and cert.key for the current switch, create the TLS KeyStore File.

If multiple switches are connected, repeat this step for each switch separately.

Step 2 Copy and convert the server.cer and cert.key files to .PEM format using the **copy** command.

Example:

```
cp cert.key sw1-ndb-privatekey.pem
cp server.cer sw1-ndb-cert.pem
```

Step 3 Concatenate the private key (sw1-ndb-privatekey.pem) and certificate file (sw1-ndb-cert.pem) into a single .PEM file using the **cat** command.

Example:

```
cat sw1-ndb-privatekey.pem sw1-ndb-cert.pem > sw1-ndb.pem
```

Step 4 Convert the .PEM file to .P12 format using the **openssl** command. Enter the export password when prompted. The password must contain at least 6 characters, for example, cisco123. The sw1-ndb.pem file is converted to a password-protected sw1-ndb.p12 file.

Example:

```
docker@docker-virtual-machine:~/TLS$openssl pkcs12 -export -out sw1-ndb.p12 -in sw1-ndb.pem
Enter Export Password: cisco123
Verifying – Enter Export Password: cisco123
Enter a password at the prompt. Use the same password that you entered in the previous Step (cisco123)
```

Step 5 Convert the sw1-ndb.p12 to a password protected Java KeyStore (tlsKeyStore) file using the **keytool** command. This command converts the sw1-ndb.p12 file to a password-protected tlsKeyStore file.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -importkeystore -srckeystore sw1-ndb.p12 -srcstoretype
 pkcs12 -destkeystore tlsKeyStore -deststoretype jks
Enter Destination Keystore password:cisco123
```

Note
By default an alias named “1” is stored in tlsKeyStore for the first switch. If the NDB controller is managing multiple switches, repeat this step for all the switches. When you add the second switch, the utility allows you to rename the first switch alias and also provides a provision to rename alias for the new switch. For example, see below.

```
keytool -importkeystore -srckeystore sw2-ndb.p12 -srcstoretype pkcs12 -destkeystore tlsKeyStore
-deststoretype jks
keytool -importkeystore -srckeystore sw3-ndb.p12 -srcstoretype pkcs12 -destkeystore tlsKeyStore
-deststoretype jks
```

Step 6 List and verify content in the java tlsKeyStore using the keytool command.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -list -v -keystore tlsKeyStore | more
```

Step 7 Convert the certificate file (sw1-ndb-cert.pem) to a Java TrustStore (tlsTrustStore) file using the **keytool** command. Enter a password when prompted to create a password protected Java TrustStore (tlsTrustStore) file. The password should be at least six characters.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -import -alias sw1 -file sw1-ndb-cert.pem -keystore
tlsTrustStore -storetype jks
Enter keystore password: cisco123
Re-enter new password: cisco123
Owner: EMAILADDRESS=myname@cisco.com, CN=localhost, OU=insbu, O=cisco, L=SJ, ST=CA, C=US
Issuer: EMAILADDRESS=myname@cisco.com, CN=localhost, OU=insbu, O=cisco, L=SJ, ST=CA, C=US
```

```
Serial number: c557f668a0dd2ca5
Valid from: Thu Jun 15 05:43:48 IST 2017 until: Sun Jun 13 05:43:48 IST 2027
Certificate fingerprints:
MD5: C2:7B:9E:26:31:7A:74:25:55:DF:A7:91:C9:5D:20:A3
SHA1: 3C:DF:66:96:72:12:CE:81:DB:AB:58:30:60:E7:CC:04:4D:DF:6D:B2
SHA256: DD:FB:3D:71:B4:B8:9E:CE:97:A3:E4:2D:D3:B6:90:CD:76:A8:5F:84:77:78:BE:49:6C:04:01:84:62:2C:2F:EB
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 0D B3 CF 81 66 4A 33 4E EF 86 7E 26 C3 50 9B 73 ....fJ3N...&.P.s
0010: 38 EF DF 40 8..@
]
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
CA:true
PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 0D B3 CF 81 66 4A 33 4E EF 86 7E 26 C3 50 9B 73 ....fJ3N...&.P.s
0010: 38 EF DF 40 8..@
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
```

Note
If a NDB controller manages multiple switches, repeat this step for all the switches to add all switch keys into the same TrustStore. For example:

```
keytool -import -alias sw2 -file sw2-ndb-cert.pem -keystore tlsTrustStore
keytool -import -alias sw3 -file sw3-ndb-cert.pem -keystore tlsTrustStore
```

Step 8 List and verify keys for multiple switches in the same tlsTrustStore using the keytool command.

Example:

```
docker@docker-virtual-machine:~/TLS$ keytool -list -v -keystore tlsTrustStore | more
```

Starting NDB with TLS

To start NDB with TLS, complete these steps:

Procedure

Step 1 Log in to the NDB server.

Step 2 Stop the NDB application, if running, using the **runndb.sh** command

Example:

```
 ./runndb.sh -stop
Controller with PID: 17426 -- Stopped!
```

Note
When onboarding a device, ensure to provide the FQDN or IP address of the device, that was provided during the certificate generation for that device.

Step 3 Copy the tlsKeystore and tlsTruststore files that you created to configuration folder of NDB (ndb/configuration).

Example:

```
cp tlskeystore /root/ndb/configuration
cp tlsTrustStore /root/ndb/configuration
```

Step 4 Start the NDB application with TLS using the **runndb.sh** script.

Example:

```
./runndb.sh -tls -tlskeystore ./configuration/tlsKeyStore -tlstruststore ./configuration/tlsTrustStore
```

Example:

To start NDB with default username (admin) and a non-default password (for example, pwd123):

```
./runndb.sh -osgiPasswordSync -tls -tlskeystore ./configuration/tlsKeyStore -tlstruststore
./configuration/tlsTrustStore
If ndb password is changed, OSGi webconsole password needs to be changed.
To set non-default OSGi webconsole password, enter ndb Admin Password [default]:
 (Type the non-default password which was set)
```

Note
To disable TLS, run the **./runndb.sh -notls** command. To disable TLS and start NDB, run the **./runndb.sh -notls -start** command. Before disabling TLS, ensure to *stop* NDB. After TLS is disabled, the port number for the devices connected to the NDB server should be changed to 80.

Configuring TLS KeyStore and TrustStore Passwords on Nexus Dashboard Data Broker

You need to configure TLS KeyStore and TrustStore passwords to enable Nexus Dashboard Data Broker to read password protected TLS KeyStore and TrustStore files. To configure TLS KeyStore and TrustStore passwords on Nexus Dashboard Data Broker , complete these steps:

Procedure

Step 1 Log in to the Nexus Dashboard Data Broker server.

Step 2 Navigate to bin directory.

Example:

```
cd ndb/bin
```

Step 3 Configure the TLS KeyStore and TrustStore passwords using the **ndb config-keystore-passwords** command.

Example:

```
./ndb config-keystore-passwords --user admin --password admin --url https://ip-address_localhost:8443
 --verbose --prompt --keystore-password keystore_password --truststore-password truststore_password
Please enter your password: <enter the NDB GUI admin password>
```

In case Nexus Dashboard Data Broker is configured with AAA (Tacacs/LDAP/Radius), and if the above command, **ndb config-keystore-passwords** fails, and you see a *401 unauthorized* error, then:

a. Go to `ndb` or `xnc` directory.

b. Stop the Nexus Dashboard Data Broker server using `./runndb.sh -stop`.

c. Enable the flag `enable.LocalUser.Authentication` by changing the value from *false* to *true* in the Nexus Dashboard Data Broker **config.ini** file.

d. Start the Nexus Dashboard Data Broker server using `./runndb.sh -start`.

e. Run the **ndb config-keystore-passwords** command again.

Note
In a HA environment, you need to run the above procedure for all the Nexus Dashboard Data Broker servers in the cluster.

After the TLS is enabled on Nexus Dashboard Data Broker , all the connections between Nexus Dashboard Data Broker server and Nexus Dashboard Data Broker switch are established using port 443. Ensure that you change device connections in Nexus Dashboard Data Broker to use port 443.

Up on successfully completing these steps, you can add nexus switch in the controller using port 443. Use FQDN of the switch to add the device to the Nexus Dashboard Data Broker controller.

You can verify the Certificate information using the WebUI Sandbox of the switch.

Generating TLS Self-Signed Certification Between WebUI Browser and NDB Server

You can secure communication between a Web browser and NDB server running in centralized mode using self-signed certificates. This section describes how to generate a self-signed certificate to secure communication between a WebUI browser and NDB application. By default Cisco NDB is shipped with default certificate which is issued to Cisco NDB and issued by Cisco NDB with default validity. You can use the **generateWebUIcertificate.sh** script under configuration folder to create self-signed certificates. For Cisco NDB releases 3.5 and earlier, these certificates are valid for 6 months. Starting with Cisco NDB release 3.6, default validity of a certificate is 6 months but you can configure the validity of a certificate.

Note You can create self-signed TLS certificates for NDB in Centralized mode only.

- Generating TLS Self-Signed Certification Between WebUI Browser and NDB Server Running in Centralized Mode

Generating TLS Self-Signed Certification Between WebUI Browser and NDB Server Running in Centralized Environment

Complete the following steps to generate TLS self-signed certification between WebUI Browser and NDB Server running in Centralized mode:

Procedure

Step 1 Log into the NDB server and change the current directory `\ndb\configuration`.

Example:

```
[root@RHEL-VM-NDB-ACI]# cd \ndb\configuration
```

Step 2 Generate the TLS self-signed certificate using the **generateWebUIcertificate.sh** script.

Example:

```
[root@RHEL-VM-NDB-ACI configuration]# ./generateWebUIcertificate.sh

**********************************
Entor Fully qualified domain name :
**********************************
NDB-browser This can be FQDN of the NDB java application as well
**************************
Enter Organizational unit :
**************************
INSBU
********************
Enter Organization :
********************
cisco
****************
Enter Location :
****************
SJ
*************
Enter State :
*************
CA
***************
Enter Country :
***************
USA
***************
Enter keypass :
***************
```

```
cisco123
*****************
Enter storepass :
******************
cisco123
************************************
Enter the validity in number of days :
************************************
365  in NDB 3.5 this script will let you to specify the certificate validity.
*************************************************************************************************************************
Below process will rename the existing key file to <old_keystore>, will generate
 a new key file. Do you want to continue (y/n) ?
*************************************************************************************************************************
y
*******************************
Self-Signed Certificate Created
*******************************
Alias name: cisco
Creation date: Jan 6, 2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=NDB-browser, OU=INSBU, O=cisco, L=SJ, ST=CA, C=USA
Issuer: CN=NDB-browser, OU=INSBU, O=cisco, L=SJ, ST=CA, C=USA
Serial number: b404be5
Valid from: Sun Jan 06 20:22:05 PST 2019 until: Mon Jan 06 20:22:05 PST 2020
Certificate fingerprints:
         MD5:  71:07:F6:4E:57:6A:08:3A:AD:06:32:B3:6C:5F:8F:52
         SHA1: 04:08:B9:D5:B7:EB:ED:E0:F9:22:49:14:FA:C6:09:39:22:32:43:A2
         SHA256:
34:D9:EB:34:0A:52:D1:4A:DD:F1:8B:14:D0:84:E4:1C:57:8B:2B:99:9B:E5:A1:4C:C7:8C:CD:AE:24:31:49:75

         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 63 8A 92 8F 6F 0F 45 BD   EE 55 C5 A8 99 3B F6 F7  c...o.E..U...;..
0010: AC FA 4A 21                                        ..J!
]
]

*******************************
Displayed the generated keystore
*******************************
*************************************************
Configured the keystore details on tomcat-server.xml
*************************************************
```

```
**************************************************************************************************
The newly generated key will used on next NDB restart. Do you want to restart NDB
 now (y/n) ?
**************************************************************************************************
y
Doesn't seem any Controller daemon is currently running
Running controller in background with PID: 13573, to connect to it please SSH to
 this host on port 2400
NDB GUI can be accessed using below URL:
[https://10.16.206.160:8443]
[https://[fe80::250:56ff:fe90:b764]:8443]
[https://10.16.206.159:8443]
[https://192.168.1.123:8443]
[https://[fe80::250:56ff:fe90:9c79]:8443]

*************
NDB Restarted
*************
```

Note
The **generateWebUIcertificate.sh** script reloads the NDB application to ensure that the browser starts using this certificate when we access NDB java application from the browser.

Step 3 Decode the generated certificate using the **keytool -list -v -keystore keystore_Name** command. Enter the store password when prompted.

Example:

```
[root@RHEL-VM-NDB-ACI configuration]# keytool -list -v -keystore keystore
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: cisco
Creation date: Jul 6, 2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=NDB-browser, OU=INSBU, O=cisco, L=SJ, ST=CA, C=USA
Issuer: CN=NDB-browser, OU=INSBU, O=cisco, L=SJ, ST=CA, C=USA
Serial number: b404be5
Valid from: Sun Jan 06 20:22:05 PST 2019 until: Mon Jan 06 20:22:05 PST 2020
Certificate fingerprints:
         MD5:  71:07:F6:4E:57:6A:08:3A:AD:06:32:B3:6C:5F:8F:52
         SHA1: 04:08:B9:D5:B7:EB:ED:E0:F9:22:49:14:FA:C6:09:39:22:32:43:A2
         SHA256:
34:D9:EB:34:0A:52:D1:4A:DD:F1:8B:14:D0:84:E4:1C:57:8B:2B:99:9B:E5:A1:4C:C7:8C:CD:AE:24:31:49:75
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 63 8A 92 8F 6F 0F 45 BD   EE 55 C5 A8 99 3B F6 F7  c...o.E..U...;..
```

```
0010: AC FA 4A 21                                        ..J!
]
]

*******************************************
*******************************************
```

Step 4 The self-signed certificates are generated in JKS format which is not compatible with the browsers. Hence, you need to convert these certificates into PKCS12 format before importing the certificate in the browser. Complete the following steps to convert JKS format certificate to PKCS12 format. Convert JKS format certificate into PKCS12 format using the **keytool** command.

Note
Ensure that you keep a copy of the original certificates before proceeding with the conversion.

Example:

```
keytool -importkeystore -srckeystore keystore -srcstorepass cisco123 -srckeypass cisco123 -destkeystore
keystore.p12 -deststoretype PKCS12 -srcalias cisco -deststorepass cisco123 -destkeypass cisco123
```

Note
The inputs in the **keytool** command should match the inputs provided during UI certificate generation.

Note
The resulting certificate file (keystore.p12) is in PKSC12 format.

Step 5 Add this certificate to Trusted Root certificate store on the browser. See help for respective Web browsers about how to add the certificate to the Trusted Root certificate store.

Generating TLS Self-Signed Certificate Between Web Browser and NDB Server Running in Embedded Mode Using Guest Shell Environment

To generate TLS self-signed certificate between Web browser and NDB server running in embedded mode using Guest Shell environment, complete the following steps.

Procedure

Step 1 Connect to Guest Shell using **guestshell** command.

Example:

```
N9K-C93108TC-EX-108# guestshell
[admin@guestshell ~]$
[admin@guestshell ~]$
```

Step 2 Change the current directory to `\ndb\configuration`.

Example:

```
[admin@guestshell ~]$ cd \ndb\configuration
```

Step 3 Generate the TLS self-signed certificate using the `/home/admin/ndb/configuration/generateWebUIcertificate.sh` script.

Example:

```
[root@RHEL-VM-NDB-ACI configuration]# ./generateWebUIcertificate.sh

**********************************
Entor Fully qualified domain name :
**********************************
NDB-browser This can be FQDN of the NDB java application as well
**************************
Enter Organizational unit :
**************************
INSBU
********************
Enter Organization :
********************
cisco
****************
Enter Location :
****************
SJ
*************
Enter State :
*************
CA
***************
Enter Country :
***************
USA
***************
Enter keypass :
***************
cisco123
*****************
Enter storepass :
*****************
cisco123
************************************
Enter the validity in number of days :
************************************
365  in NDB 3.5 this script will let you to specify the certificate validity.
************************************************************************************************************************************
Below process will rename the existing key file to <old_keystore>, will generate
 a new key file. Do you want to continue (y/n) ?
************************************************************************************************************************************
y
******************************
Self-Signed Certificate Created
******************************
Alias name: cisco
```

```
Creation date: Jan 6, 2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=NDB-browser, OU=INSBU, O=cisco, L=SJ, ST=CA, C=USA
Issuer: CN=NDB-browser, OU=INSBU, O=cisco, L=SJ, ST=CA, C=USA
Serial number: b404be5
Valid from: Sun Jan 06 20:22:05 PST 2019 until: Mon Jan 06 20:22:05 PST 2020
Certificate fingerprints:
         MD5:  71:07:F6:4E:57:6A:08:3A:AD:06:32:B3:6C:5F:8F:52
         SHA1: 04:08:B9:D5:B7:EB:ED:E0:F9:22:49:14:FA:C6:09:39:22:32:43:A2
         SHA256:
34:D9:EB:34:0A:52:D1:4A:DD:F1:8B:14:D0:84:E4:1C:57:8B:2B:99:9B:E5:A1:4C:C7:8C:CD:AE:24:31:49:75

         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 63 8A 92 8F 6F 0F 45 BD   EE 55 C5 A8 99 3B F6 F7  c...o.E..U...;..
0010: AC FA 4A 21                                        ..J!
]
]

*******************************
Displayed the generated keystore
*******************************
*************************************************
Configured the keystore details on jetty-ssl-context.xml
*************************************************
*********************************************************************************************
The newly generated key will used on next NDB restart. Do you want to restart NDB
 now (y/n) ?
*********************************************************************************************
n
*************
The newly generated key will be used on the next NDB restart.
*************
```

Note
Manually reboot guestshell using the **guestshell reboot** command to ensure that the browser starts using this certificate when you access NDB java application from the browser.

Step 4 Decode the generated certificate using the **keytool -list -v -keystore keystore_Name** command. Enter the store password when prompted.

Example:

```
[root@RHEL-VM-NDB-ACI configuration]#  keytool -list -v -keystore keystore
Enter keystore password:
```

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: cisco
Creation date: Jul 6, 2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=NDB-browser, OU=INSBU, O=cisco, L=SJ, ST=CA, C=USA
Issuer: CN=NDB-browser, OU=INSBU, O=cisco, L=SJ, ST=CA, C=USA
Serial number: b404be5
Valid from: Sun Jan 06 20:22:05 PST 2019 until: Mon Jan 06 20:22:05 PST 2020
Certificate fingerprints:
         MD5:  71:07:F6:4E:57:6A:08:3A:AD:06:32:B3:6C:5F:8F:52
         SHA1: 04:08:B9:D5:B7:EB:ED:E0:F9:22:49:14:FA:C6:09:39:22:32:43:A2
         SHA256:
34:D9:EB:34:0A:52:D1:4A:DD:F1:8B:14:D0:84:E4:1C:57:8B:2B:99:9B:E5:A1:4C:C7:8C:CD:AE:24:31:49:75
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 63 8A 92 8F 6F 0F 45 BD   EE 55 C5 A8 99 3B F6 F7  c...o.E..U...;..
0010: AC FA 4A 21                                        ..J!
]
]

*******************************************
*******************************************
```

Step 5 The self-signed certificates are generated in JKS format which is not compatible with the browsers. You need to convert these certificates into PKCS12 format before importing the certificate in the browser. Complete the following steps to convert JKS format certificate to PKCS12 format. Convert JKS format certificate into PKCS12 format using the **keytool** command.

Note
Ensure that you keep a copy of the original certificates before proceeding with the conversion.

Example:

```
keytool -importkeystore -srckeystore keystore -srcstorepass cisco123 -srckeypass cisco123 -destkeystore
keystore.p12 -deststoretype PKCS12 -srcalias cisco -deststorepass cisco123 -destkeypass cisco123
```

Note
The inputs in the **keytool** command should match the inputs provided during UI certificate generation.

Note
The resulting certificate file (keystore.p12) is in PKSC12 format.

Step 6 Upload the CA certificate into Trusted Root certificate store of Web browser. See help for respective Web browsers about how to add the certificate to the Trusted Root certificate store. Use the password that you created while creating the certificate when prompted while uploading the certificate to the Web browser.

Step 7 Restart the Guest Shell to restart the NDB.

Generating TLS 3rd Party Certification Between WebUI Browser and NDB Server

You can secure communication between a Web browser and NDB server running in centralized mode. This section describes how to generate CA certificates, convert the certificates into JKS format, and upload the certificates into a Web browser. To generate a CA certificate, you need to generate a Certificate Signing Request (CSR) and send it to a Certificate issuing authority (CA). You can use an open source tool to generate a CSR.

- Generating TLS 3rd Party Certification Between WebUI Browser and NDB Server Running in Centralized Mode

Generating TLS 3rd Party Certification Between WebUI Browser and NDB Server Running in Centralized Mode

Complete these steps to generate TLS 3rd party certification between WebUI browser and NDB server running in centralized mode:

Procedure

Step 1 Generate Certificate Signing Request (CSR) using the **openssl req** command.

Example:

```
[root@NDB-server ~]# openssl req -newkey rsa:2048 -sha256 -keyout ndb-server.key -keyform PEM -out
 ndb-server.req -outform PEM
Generating a 2048 bit RSA private key
...+++
.....................................................+++
writing new private key to 'ndb-server.key'
Enter PEM pass phrase:  cisco123
Verifying - Enter PEM pass phrase:  cisco123
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:CA
Locality Name (eg, city) [Newbury]:SJ
```

```
Organization Name (eg, company) [My Company Ltd]:cisco
Organizational Unit Name (eg, section) []:insbu
Common Name (eg, your name or your server's hostname) []:ndb-server.cisco.com
Email Address []:chburra@cisco.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:cisco123
An optional company name []:cisco123


[root@NDB-server ~]# ls
ndb-server.req   ndb-server.key
```

Note
The ndb-server.req (CSR) file is submitted to the certificate issuing authority (CA).

Note
You need to use the same information when exporting the CA provided certificate into browser. The CSR file, cert.req, is submitted to CA.

Step 2 To verify or view the CSR request, use the **openssl req** command.

Example:

```
[root@NDB-server ~]# openssl req -noout -text -in ndb-server.req
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=US, ST=CA, L=SJ, O=cisco, OU=insbu,
CN=ndb-server.cisco.com/emailAddress=chburra@cisco.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b5:30:75:e8:c8:5f:05:3b:0e:4f:aa:00:d9:64:
                    8d:bf:b2:80:20:56:c3:be:b0:4c:e0:52:e5:be:d8:
                    d2:74:85:4e:8a:ba:d3:1e:30:76:bf:e5:de:7d:51:
                    11:79:8e:bc:96:38:7a:23:5a:26:31:50:50:fa:29:
                    44:ab:56:b6:0d:41:38:ba:d1:d5:b4:e3:ba:a3:6c:
                    4a:35:73:27:d9:fd:5c:4b:21:85:1a:f9:4d:b0:9e:
                    f3:ae:ce:49:98:ef:a2:f8:11:ab:bd:7e:64:ee:68:
                    68:19:6e:8f:3c:54:30:0f:28:01:13:b0:3d:34:b8:
                    f9:f5:cc:4a:84:d8:e5:d2:27:47:cc:83:76:92:ad:
                    92:62:f3:a3:35:be:14:ce:38:af:2a:c5:2e:fa:b8:
                    31:6b:71:cd:56:00:1f:0d:cc:b0:f8:fc:b0:52:91:
                    f8:9c:cf:45:13:c9:b5:86:fa:30:dd:88:78:01:15:
                    fb:5c:c9:6f:5b:b7:80:28:6c:86:54:c0:f2:5f:35:
                    70:82:49:5c:79:1c:f2:23:dd:50:d5:47:12:37:a3:
                    3f:f9:1d:90:8f:c0:e8:18:09:2e:66:8d:c3:72:17:
                    7f:7d:27:da:b1:cc:26:2d:8c:6b:ee:c5:e8:b5:78:
                    31:7c:bb:ba:6d:2c:e5:a3:29:7e:c1:4a:93:19:ed:
                    9a:e7
```

```
            Exponent: 65537 (0x10001)
        Attributes:
            unstructuredName         :cisco123
            challengePassword        :cisco123
    Signature Algorithm: sha256WithRSAEncryption
        9c:9a:51:e0:1d:e4:0b:8f:c1:c6:f5:e0:d2:f6:30:0e:18:af:
        a7:b2:a4:4a:57:d7:07:44:cd:9c:fa:2d:0e:8b:c9:31:5b:16:
        6b:84:42:0b:ed:06:5c:ed:30:d8:9b:ee:5d:79:f4:8a:e3:52:
        3c:b3:4a:eb:6c:22:a2:f4:35:80:28:3a:67:62:7f:5f:dc:80:
        e0:74:f0:3c:39:26:39:3a:76:6a:6a:98:e9:68:f9:b7:58:bf:
        e7:44:2e:e7:73:0a:9c:62:28:b2:c6:09:41:81:b2:53:46:14:
        e6:e4:dc:ca:90:81:5a:5e:dc:1b:dc:36:2c:86:5f:37:29:4c:
        b0:ee:85:2b:34:f2:82:8a:d4:fc:a0:ce:10:e4:44:4e:d0:7a:
        37:6d:3e:f9:ff:a1:19:8c:db:06:bf:be:87:57:a1:cb:05:15:
        0b:9f:6c:8b:c2:ad:22:25:10:f0:4d:0f:4d:b7:be:71:87:f7:
        85:24:e7:2d:f9:59:86:1a:b7:88:57:16:93:31:1f:d7:e5:07:
        42:77:00:f9:ac:44:3b:6c:35:0f:80:5d:00:6f:ea:be:fe:e7:
        28:53:0c:6b:5f:0c:76:bf:8c:a7:60:57:63:05:06:ff:ac:3d:
        f1:63:54:d0:d0:13:44:b1:e9:53:6b:32:11:e2:83:26:04:f5:
        23:67:6b:de
```

Step 3 The private key, ndb-server.key, is secured with the passphrase. You need to unencrypt the certificate private key. Unencrypt the private key using the **openssl rsa** command.

Example:

```
[root@NDB-server ~]# cp ndb-server.key  ndb-server.keybkp
[root@NDB-server ~]# rm ndb-server.key
 [root@NDB-server ~]# openssl rsa -in ndb-server.keybkp -out ndb-server.key
Enter pass phrase for ndb-server.keybkp: cisco123
writing RSA key
```

Note
The ndb-server.req file data needs to be submitted to 3rd party certification authority. Follow the relevant procedures and get the certificate files.

Depending on the tier of the CA you choose, you can get up to three certificates (certificate chain) for each CSR. This means you get three certificates (root, intermediate and domain) from CA for each NDB switch. You need to check with CA to identify each type of certificate. Certificate naming convention might be different for different certifying authorities. For example: qvrca2.cer (root), hydsslg2.cer (intermediate), ndb-server.cisco.com-39891.cer (domain).

Certificates are mostly shared in .PEM file format.

Step 4 Create a single certificate file from the three certificate files using the cat command. The concatenation should be done in the following order, domain certificate, root certificate, and intermediate certificate. Syntax for cat command: **cat** *domain certificate root certificate intermediate certificate* > *ndb-server.cer*.

Example:

```
[root@NDB-server ~]# cat ndb-server.cisco.com-39891.cer qvrca.cer hydsslg2.cer > ndb-server.cer
```

Step 5 Edit the newly created server.cer file to separate the concatenated END and BEGIN lines. Do not delete anything in the file.

Example:

```
-----END CERTIFICATE----------BEGIN CERTIFICATE-----
```

```
////// Modify the above line like this by adding a line feed between the two.
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
```

Step 6 Create the TLS NDB Server Keystore file using the ndb-server.cer and ndb-server.key files. Copy the files to switch using the copy command.

Example:

```
cp ndb-server.key ndb-server-ndb-privatekey.pem
      cp ndb-server.cer ndb-server-ndb-cert.pem
```

Step 7 Combine the private key and certificate file into a single .PEM file using the cat command.

Example:

```
cat ndb-server-ndb-privatekey.pem  ndb-server-ndb-cert.pem > ndb-server-ndb.pem
```

Step 8 CA provides certificates in PEM format and extension of the certificate is .pem. You need to convert the PEM format certificate to PKCS12 format. Convert the PEM file, ndb-server-ndb.pem, to .P12 file format using the **openssl pkcs12** command. Enter the export password when prompted. The password must contain at least 6 characters, for example, cisco123. The ndb-server-ndb.pem file is converted to a password-protected ndb-server-ndb.p12 file.

Example:

```
[root@NDB-server ~]# openssl pkcs12 -export -out ndb-server-ndb.p12 -in ndb-server-ndb.pem
Enter Export Password:□cisco123
Verifying - Enter Export Password:□cisco123
```

Step 9 Convert the ndb-server-ndb.p12 to a password protected Java KeyStore (ndb-server-keystore) file using the **keytool** command. This command converts the sw1-ndb.p12 file to a password-protected ndb-server-keystore file. Create a new password for the destination JKS store and enter the source keystore password when prompted.

Example:

```
[root@NDB-server ~]# .(relativePath)/keytool -importkeystore -srckeystore ndb-server-ndb.p12
-srcstoretype pkcs12 -destkeystore ndb-server-keystore -deststoretype jks
Enter destination keystore password:□cisco123
Re-enter new password:□cisco123
Enter source keystore password:--cisco123
Entry for alias 1 successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or
 cancelled
[root@NDB-server ~]#
```

Step 10 List and verify content in the java tlsKeyStore using the **keytool** command.

Example:

```
[root@NDB-server ~]#.(relativePath)/keytool -list -v -keystore ndb-server-keystore
```

Step 11 Copy the ndb-server-keystore file to the */ndb/configuration* folder.

Step 12 Configure `jetty-ssl-context.xml` (stored in */ndb/configuration/*etc) with key store password that was provided while generating the certificate. You can use VI editor and edit the following lines with KeyStorePath, KeyStorePassword, TrustStorePath, TrustStorePassword.

Example:

```
<Set name="KeyStorePath"><Property name="jetty.base" default="." />/<Property
name="jetty.sslContext.keyStorePath" deprecated="jetty.keystore"
default="configuration/ndb-server-keystore"/></Set>
```

```
<Set name="KeyStorePassword"><Property name="jetty.sslContext.keyStorePassword"
deprecated="jetty.keystore.password" default="cisco123"/></Set>

<Set name="KeyManagerPassword"><Property name="jetty.sslContext.keyManagerPassword"
deprecated="jetty.keymanager.password" default="cisco123"/></Set>
<Set name="TrustStorePath"><Property name="jetty.base" default="." />/<Property
name="jetty.sslContext.trustStorePath" deprecated="jetty.truststore"
default="configuration/ndb-server-keystore"/></Set>

<Set name="TrustStorePassword"><Property name="jetty.sslContext.trustStorePassword"
deprecated="jetty.truststore.password" default="cisco123"/></Set>
```

Step 13 Restart the NDB.

Step 14 Upload the CA certificate into Trusted Root certificate store of Web browser. See help for respective Web browsers about how to add the certificate to the Trusted Root certificate store. Use the password that you created while creating the certificate when prompted while uploading the certificate to the Web browser.

Generating TLS 3rd Party Certificate Between Web Browser and NDB Server Running in Embedded Mode Using Guest Shell Environment

To generate TLS 3rd party certificate between Web browser and NDB server running in embedded mode using guest shell environment, complete the following steps.

Procedure

Step 1 Enable bash-shell feature on the switch using the feature command.

Example:

```
N9396TX-116(config)# feature bash-shell
```

Step 2 Enter bash-shell mode on the switch using the run command.

Example:

```
N9396TX-116(config)# run bash
bash-4.2$
```

Step 3 Generate Certificate Signing Request (CSR) using the **openssl req** command. Enter the required information when prompted.

Example:

```
bash-4.2$ openssl req -newkey rsa:2048 -sha256 -keyout ndb-server.key -keyform PEM -out ndb-server.req
 -outform PEM
Generating a 2048 bit RSA private key
...+++
.............................................+++
writing new private key to 'ndb-server.key'
Enter PEM pass phrase:  cisco123
Verifying - Enter PEM pass phrase:  cisco123
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:CA
Locality Name (eg, city) [Newbury]:SJ
Organization Name (eg, company) [My Company Ltd]:cisco
Organizational Unit Name (eg, section) []:insbu
Common Name (eg, your name or your server's hostname) []:ndb-server.cisco.com
Email Address []:chburra@cisco.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:cisco123
An optional company name []:cisco123



bash-4.2$ ls

ndb-server.req   ndb-server.key
```

Note
The openssl command creates a privsate key, ndb-server.key, and a certificate signing request file, ndb-server.req. The ndb-server.req (CSR) file is submitted to the certificate issuing authority (CA).

Note
You need to use the same information when exporting the CA provided certificate into browser. The CSR file, cert.req, is submitted to CA.

Step 4 To view the content or verify the CSR request, use the **openssl req** command.

Example:

```
bash-4.2$ openssl req -noout -text -in ndb-server.req
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=US, ST=CA, L=SJ, O=cisco, OU=insbu,
CN=ndb-server.cisco.com/emailAddress=chburra@cisco.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b5:30:75:e8:c8:5f:05:3b:0e:4f:aa:00:d9:64:
                    8d:bf:b2:80:20:56:c3:be:b0:4c:e0:52:e5:be:d8:
                    d2:74:85:4e:8a:ba:d3:1e:30:76:bf:e5:de:7d:51:
                    11:79:8e:bc:96:38:7a:23:5a:26:31:50:50:fa:29:
                    44:ab:56:b6:0d:41:38:ba:d1:d5:b4:e3:ba:a3:6c:
                    4a:35:73:27:d9:fd:5c:4b:21:85:1a:f9:4d:b0:9e:
                    f3:ae:ce:49:98:ef:a2:f8:11:ab:bd:7e:64:ee:68:
                    68:19:6e:8f:3c:54:30:0f:28:01:13:b0:3d:34:b8:
                    f9:f5:cc:4a:84:d8:e5:d2:27:47:cc:83:76:92:ad:
                    92:62:f3:a3:35:be:14:ce:38:af:2a:c5:2e:fa:b8:
                    31:6b:71:cd:56:00:1f:0d:cc:b0:f8:fc:b0:52:91:
                    f8:9c:cf:45:13:c9:b5:86:fa:30:dd:88:78:01:15:
                    fb:5c:c9:6f:5b:b7:80:28:6c:86:54:c0:f2:5f:35:
                    70:82:49:5c:79:1c:f2:23:dd:50:d5:47:12:37:a3:
                    3f:f9:1d:90:8f:c0:e8:18:09:2e:66:8d:c3:72:17:
                    7f:7d:27:da:b1:cc:26:2d:8c:6b:ee:c5:e8:b5:78:
                    31:7c:bb:ba:6d:2c:e5:a3:29:7e:c1:4a:93:19:ed:
                    9a:e7
                Exponent: 65537 (0x10001)
```

```
        Attributes:
            unstructuredName         :cisco123
            challengePassword        :cisco123
    Signature Algorithm: sha256WithRSAEncryption
        9c:9a:51:e0:1d:e4:0b:8f:c1:c6:f5:e0:d2:f6:30:0e:18:af:
        a7:b2:a4:4a:57:d7:07:44:cd:9c:fa:2d:0e:8b:c9:31:5b:16:
        6b:84:42:0b:ed:06:5c:ed:30:d8:9b:ee:5d:79:f4:8a:e3:52:
        3c:b3:4a:eb:6c:22:a2:f4:35:80:28:3a:67:62:7f:5f:dc:80:
        e0:74:f0:3c:39:26:39:3a:76:6a:6a:98:e9:68:f9:b7:58:bf:
        e7:44:2e:e7:73:0a:9c:62:28:b2:c6:09:41:81:b2:53:46:14:
        e6:e4:dc:ca:90:81:5a:5e:dc:1b:dc:36:2c:86:5f:37:29:4c:
        b0:ee:85:2b:34:f2:82:8a:d4:fc:a0:ce:10:e4:44:4e:d0:7a:
        37:6d:3e:f9:ff:a1:19:8c:db:06:bf:be:87:57:a1:cb:05:15:
        0b:9f:6c:8b:c2:ad:22:25:10:f0:4d:0f:4d:b7:be:71:87:f7:
        85:24:e7:2d:f9:59:86:1a:b7:88:57:16:93:31:1f:d7:e5:07:
        42:77:00:f9:ac:44:3b:6c:35:0f:80:5d:00:6f:ea:be:fe:e7:
        28:53:0c:6b:5f:0c:76:bf:8c:a7:60:57:63:05:06:ff:ac:3d:
        f1:63:54:d0:d0:13:44:b1:e9:53:6b:32:11:e2:83:26:04:f5:
        23:67:6b:de
```

Step 5 The private key, ndb-server.key, is secured with the passphrase. You need to unencrypt the certificate private key. Unencrypt the private key using the **openssl rsa** command.

Example:

```
bash-4.2$ cp ndb-server.key  ndb-server.keybkp
bash-4.2$ rm ndb-server.key
bash-4.2$ openssl rsa -in ndb-server.keybkp -out ndb-server.key
Enter pass phrase for ndb-server.keybkp: cisco123
writing RSA key
```

Note
Depending on the tier of the CA you choose, you can get up to three certificates (certificate chain) for each CSR. This means you get three certificates (root, intermediate and domain) from CA for each NDB switch. You need to check with CA to identify each type of certificate. Certificate naming convention might be different for different certifying authorities. For example: qvrca2.cer (root), hydsslg2.cer (intermediate), ndb-server.cisco.com-39891.cer (domain).

Certificates are mostly shared in .PEM file format.

Step 6 Create a single certificate file from the three certificate files using the cat command. The concatenation should be done in the following order, domain certificate, root certificate, and intermediate certificate. Syntax for cat command: **cat** *domain certificate root certificate intermediate certificate > ndb-server.cer*.

Example:

```
bash-4.2$ cat ndb-server.cisco.com-39891.cer qvrca.cer hydsslg2.cer > ndb-server.cer
```

Step 7 Edit the newly created server.cer file to separate the concatenated END and BEGIN lines. Do not delete anything in the file.

Example:

```
-----END CERTIFICATE----------BEGIN CERTIFICATE-----


////// Modify the above line like this by adding a line feed between the two.
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
```

Step 8 Create the TLS NDB Server Keystore file using the ndb-server.cer and ndb-server.key files. Copy the files to switch using the **copy** command.

Example:

```
cp ndb-server.key ndb-server-ndb-privatekey.pem
      cp ndb-server.cer ndb-server-ndb-cert.pem
```

Step 9 Combine the private key and certificate file into a single .PEM file using the **cat** command.

Example:

```
cat ndb-server-ndb-privatekey.pem  ndb-server-ndb-cert.pem > ndb-server-ndb.pem
```

Step 10 CA provides certificates in PEM format and extension of the certificate is .pem. You need to convert the PEM format certificate to PKCS12 format. Convert the PEM file, ndb-server-ndb.pem, to .P12 file format using the **openssl pkcs12** command. Enter the export password when prompted. The password must contain at least 6 characters, for example, cisco123. The ndb-server-ndb.pem file is converted to a password-protected ndb-server-ndb.p12 file.

Example:

```
bash-4.2$ openssl pkcs12 -export -out ndb-server-ndb.p12 -in ndb-server-ndb.pem
Enter Export Password:□cisco123
Verifying - Enter Export Password:□cisco123
```

Step 11 Copy the certificate file to the NDB configuration folder.

Example:

```
bash-4.2$ sudo cp ndb-server-ndb.p12
/isan/vdc_1/virtual-instance/guestshell+/rootfs/usr/bin/ndb/configuration/
```

Step 12 Exit the bash shell mode using the **exit** command.

Example:

```
bash-4.2$ exit
exit
N9396TX-116#
```

Step 13 Connect to guest shell using guestshell command.

Example:

```
N9396TX-116# guestshell
[admin@guestshell ~]$
```

Step 14 Change current directory to `ndb/configuration`.

Example:

```
[admin@guestshell ~]$ cd ndb/configuration
```

Step 15 Convert the ndb-server-ndb.p12 to a password protected Java KeyStore (ndb-server-keystore) file using the **keytool** command. This command converts the ndb-server-ndb.p12 file to a password-protected ndb-server-keystore file. Create a new password for the destination JKS store and enter the source keystore password when prompted.

Example:

```
[admin@guestshell configuration]$ keytool -importkeystore -srckeystore ndb-server-ndb.p12
-srcstoretype pkcs12 -destkeystore ndb-server-keystore -deststoretype jks
Enter destination keystore password: □cisco123
Re-enter new password: □cisco123
Enter source keystore password: □cisco123
Entry for alias 1 successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled
```

Step 16 List and verify content in the java tlsKeyStore using the **keytool** command.

Example:

```
[admin@guestshell configuration]$ keytool -list -v -keystore ndb-server-keystore
```

Step 17 Configure `jetty-ssl-context.xml` (stored in ndb/etc folder) with key store password that was provided while generating the certificate. You can use VI editor and edit the following lines with keystore and keystorepass.

Example:

```
<Set name="KeyStorePath"><Property name="jetty.base" default="." />/<Property
name="jetty.sslContext.keyStorePath" deprecated="jetty.keystore"
default="configuration/ndb-server-keystore"/></Set>
<Set name="KeyStorePassword"><Property name="jetty.sslContext.keyStorePassword"
deprecated="jetty.keystore.password" default="cisco123"/></Set>

<Set name="KeyManagerPassword"><Property name="jetty.sslContext.keyManagerPassword"
deprecated="jetty.keymanager.password" default="cisco123"/></Set>

<Set name="TrustStorePath"><Property name="jetty.base" default="." />/<Property
name="jetty.sslContext.trustStorePath" deprecated="jetty.truststore"
default="configuration/ndb-server-keystore"/></Set>

<Set name="TrustStorePassword"><Property name="jetty.sslContext.trustStorePassword"
deprecated="jetty.truststore.password" default="cisco123"/></Set>
```

Step 18 Upload the CA certificate into Trusted Root certificate store of Web browser. See help for respective Web browsers about how to add the certificate to the Trusted Root certificate store. Use the password that you created while creating the certificate when prompted while uploading the certificate to the Web browser.

Step 19 Use **guestshell disable** followed by **guestshell enable** to restart NDB.