TLS 証明書、KeyStore およびトラストストア ファイルの管理

この章では、TLS 証明書とトラストストア ファイルを生成するための情報と手順について説明します。

リリース 3.10.1 以降、Cisco Nexus Data Broker (NDB) は Cisco Nexus Dashboard Data Broker に名前が変更されました。ただし、GUI およびインストール フォルダ構造と対応させるため、一部の NDB のインスタンスがこのドキュメントには残されています。NDB/ Nexus Data Broker/ Nexus Dashboard Data Brokerという記述は、相互に交換可能なものとして用いられています。

NDB サーバーと NDB スイッチの間で NXAPI を使用する TLS 自己署名証明書を生成する

このセクションでは、NDB サーバーと NDB スイッチの間で TLS 自己署名証明書を生成する方法について説明します。TLS を有効にするには、スイッチごとに証明書とキーを生成する必要があります。NDB スイッチと NDB サーバー間の TLS 通信では、ポート 443 のみを使用します。

NDB サーバーと NDB スイッチの間で NXAPI を使用する TLS 自己署名証明書を生成するには、次の手順を実行します。

- Nexus Dashboard Data Broker での TLS KeyStore と TrustStore パスワードの構成 （6 ページ）

（注） TLS を構成した後で、ポート 80 を使用して通信するようにコントローラーを構成することはできません。

自己署名証明書とキーの生成

このセクションでは、自己署名証明書とキーを生成する方法について説明します。

始める前に

スイッチにドメイン名が構成されていることを確かめるため、**ip domain-name** コマンドを使用して、NDB スイッチごとに、完全修飾ドメイン名（FQDN）が機能することを確認します。次に例を示します。

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

スイッチの FQDN は N9K-117.cisco.com に設定されています。

ステップ 1 サーバにログインします。

ステップ 2 **openssl req** コマンドを使用して、秘密キーと自己署名証明書を生成します。

例：

```
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
```

（注） 複数のスイッチがある場合、各スイッチに対して証明書ファイルと秘密キーを生成します。

このコマンドは、証明書ファイル（sw1-ca.pem）および秘密キー（sw1-ca.key）を生成します。

ステップ 3 NDB スイッチにログインします。

ステップ 4 証明書ファイル sw1-ca.pem とキー ファイル sw1-ca.key をスイッチにコピーします。**copy** コマンドを使用します。

例：

```
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)...
```

（注） 複数のスイッチをお持ちの場合、すべてのスイッチに対してこの手順を繰り返します。

ステップ 5 スイッチに証明書ファイル sw1-ca.pem とキーファイル sw1-ca.key を設定します。**nxapi** コマンドを使用します。

例：

```
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)#
```

（注） 複数のスイッチがある場合は、各スイッチに対応する証明書と秘密キーを構成します。

ステップ 6 **nxapi certificate** コマンドを使用して、スイッチの自己署名証明書を有効にします。

例：

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

（注） スイッチで自己署名証明書を有効にするときにエラーが生じないことを確認します。

ステップ 7 サーバにログインします。

ステップ 8 **copy** コマンドを使用して、sw1-ca.key および sw1-ca.pem ファイルをコピーし、.PEM 形式に変換します。

例：

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

ステップ 9 **cat** コマンドを使用して、秘密キーと証明書ファイルを連結します。

例：

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

ステップ 10 **openssl** コマンドを使用して、.pem ファイルを .p12 ファイル形式に変換します。パスワードで保護された .p12 証明書ファイルを作成するように指示メッセージが表示されたら、エクスポート パスワードを入力します。

例：

```
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)
```

ステップ 11 **keytool** コマンドを使用して、sw1-ndb.p12 をパスワード保護された Java キーストア（tlsKeyStore）ファイルに変換します。インストールされている java ディレクトリの jre/bin を使用します。

例：

```
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.
```

（注） デフォルトでは、「1」というエイリアスが最初のスイッチの tlsKeyStore に保存されます。NDB コントローラが複数のスイッチを管理している場合は、すべてのスイッチに対してこの手順を繰り返します。2番目のスイッチを追加すると、ユーティリティによって最初のスイッチのエイリアス名を変更できるようになります。2番目のスイッチのエイリアス名を変更することもできます。以下に示す例を参照してください。

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

ステップ 12 **keytool** コマンドを使用して、java tlsKeyStore のコンテンツをリストして検証します。

例：

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

次のタスク

次のタスク、*TLS TrustStore* ファイルの作成に進みます。

TLS TrustStore ファイルの作成

TrustStoreは、1つ以上のスイッチに対して生成された自己署名証明書から作成されます。このファイルはコントローラ内の 1 つ以上のスイッチの証明書を保持します。このセクションでは、自己署名証明書とキーの生成セクションで作成した自己署名証明書を使用して TrustStore を作成する方法について説明します。コントローラに複数のスイッチがある場合、各スイッチには個別の証明書ファイルがあります（たとえば、sw1-ndb-cert.pem、sw2-ndb-cert.pem）。

ステップ 1 サーバにログインします。

ステップ 2 **keytool** コマンドを使用して、証明書ファイル（たとえば、sw1-ndb-cert.pem）を Java TrustStore（tlsTrustStore）ファイルに変換します。パスワードで保護された Java TrustStore（tlsTrustStore）ファイルを作成するためにパスワードを求められたら、入力します。パスワードは 6 文字以上にする必要があります。java ディレクトリにインストールされている jre/bin を使用します。

例：

```
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)
```

（注） NDB コントローラが複数のスイッチを管理している場合は、すべてのスイッチに対してこの手順を繰り返して、すべてのスイッチ キーを同じ TrustStore に追加します。次に例を示します。

```
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.
```

ステップ 3 **keytool** コマンドを使用して、同じ tlsTrustStore 内の複数のスイッチのキーを一覧表示して確認します。

例：

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

TLS を使用した NDB の開始

TLS を使用して NDB を開始するには、次の手順を実行します。

ステップ 1 NDB サーバーにログインします。

ステップ 2 **runndb.sh** コマンドを使用して、NDB アプリケーションを停止します（実行中の場合）。

例：

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

ステップ 3 作成した tlsKeystore および tlsTruststore ファイルを NDB の構成フォルダ（ndb/configuration）にコピーします。

例：

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

ステップ 4 **runndb.sh** スクリプトを使用して、TLS で NDB アプリケーションを開始します。

例：

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

例：

デフォルトのユーザー名（admin）とデフォルト以外のパスワード（たとえば、pwd123）で NDB を起動するには：

```
./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)
```

（注） TLS を無効にするには、**./runndb.sh -notls** コマンドを実行します。TLS を無効にして NDB を開始するには、**./runndb.sh -notls -start** コマンドを実行します。TLS を無効にする前に、必ず NDB を停止してください。TLS を無効にした後、NDB サーバーに接続されているデバイスのポート番号を 80 に変更する必要があります。

Nexus Dashboard Data Broker での TLS KeyStore と TrustStore パスワードの構成

Nexus Dashboard Data Broker がパスワードで保護された TLS KeyStore と TrustStore のファイルを読み取れるようにするには、TLS KeyStore と TrustStore のパスワードを構成する必要があります。Nexus Dashboard Data Broker で TLS KeyStore と TrustStore のパスワードを構成するには、次の手順を実行します。

ステップ 1 Nexus Dashboard Data Broker サーバーにログインします。

ステップ 2 bin ディレクトリに移動します。

例：

```
cd ndb/bin
```

ステップ 3 **ndb config-keystore-passwords** コマンドを使用して、TLS KeyStore と TrustStore のパスワードを構成します。

例：

```
./ndb config-keystore-passwords --user admin --password admin --url https://ip-address_localhost:8443
 --verbose --prompt --keystore-password keystore_password --truststore-password truststore_password
```

Nexus Dashboard Data Broker が AAA（Tacacs/LDAP/Radius）で構成されており、上記のコマンドで **ndb config-keystore-passwords** が失敗し、*401 unauthorized* エラーが表示された場合：

1. `ndb` または `xnc` ディレクトリに移動します。
2. `./runndb.sh -stop` を使用して、Nexus Dashboard Data Broker サーバーを停止します。
3. フラグ `enable.LocalUser.Authentication` を *false* から *true* に変更して、有効にします。このフラグは Nexus Dashboard Data Broker の **config.ini** ファイルにあります。
4. `./runndb.sh -start` を使用して、Nexus Dashboard Data Broker サーバーを起動します。
5. **ndb config-keystore-passwords** コマンドを再度実行します。

（注） HA 環境では、クラスタ内のすべての Nexus Dashboard Data Broker サーバーに対して上記の手順を実行する必要があります。

Nexus Dashboard Data Broker で TLS を有効にすると、Nexus Dashboard Data Broker サーバーと Nexus Dashboard Data Broker スイッチ間のすべての接続がポート 443 を使用して確立されます。ポート 443 を使用するように Nexus Dashboard Data Broker のデバイス接続を変更してください。

これらの手順を正常に完了すると、ポート 443 を使用してコントローラに Nexus スイッチを追加できます。スイッチの FQDN を使用して、デバイスを Nexus Dashboard Data Broker コントローラに追加します。

スイッチの WebUI Sandbox を使用して証明書情報を確認できます。

NDB サーバーと NDB スイッチ間での NXAPI 用の TLS サードパーティ証明書の生成

このセクションでは、NDB サーバーと NDB スイッチの間で TLS サードパーティ証明書を生成する方法について説明します。ネットワーク内のスイッチごとに個別の証明書とキーを要求する必要があります。NDB サーバーと NDB スイッチの間で TLS 通信は、ポート 443 のみを使用します。

NDB サーバーと NDB スイッチの間で NXAPI 用の TLS サードパーティ証明書を生成するには、次の手順を実行します。

- 認証局から証明書を取得する
- NDB コントローラの TLS キーストアとトラストストア ファイルの作成

- TLS を使用した NDB の開始
- Nexus Dashboard Data Broker での TLS KeyStore と TrustStore パスワードの構成

（注） 両方のセクションのすべての手順を実行して、コントローラとスイッチ間の TLS での通信が正常に行われるようにします。

認証局から証明書を取得する

2 つの方法で認証局（CA）から証明書を取得できます。秘密キーと証明書の両方に対して CA に直接アプローチすることができます。CA は、発行元 CA の署名が付された公開キーを含む証明書を生成し、また申請者に代わって秘密キーを生成します。

もう 1 つのアプローチでは、openssl などのツールを使用して秘密キーを生成したうえで、証明書発行機関への証明書署名要求（CSR）を生成します。CA は、CSR のユーザー ID 情報を使用し、公開キーを使用して証明書を生成します。

始める前に

スイッチの完全修飾ドメイン名（FQDN）として機能する各 NDB スイッチに対して **ip domain-name** コマンドを使用して、スイッチにドメイン名が設定されていることを確認します。次に例を示します。

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

スイッチの FQDN は N9K-117.cisco.com に設定されています。

ステップ 1 サーバにログインします。

ステップ 2 openssl コマンドを使用して、秘密キー（cert.key）と証明書署名要求（cert.req）を生成します。

例：

```
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
```

ステップ 3 openssl コマンドを使用して CSR を確認します。

例：

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

ステップ 4 秘密キーは、セキュリティ パスフレーズを使用して生成されます。秘密キーの暗号化を解除する必要が生じるかもしれません。秘密鍵からパスフレーズを削除するには、openssl コマンドを使用します。

例：

```
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
```

（注） この手順を繰り返して、すべてのスイッチの秘密鍵からパスフレーズを削除します。

（注） 選択する CA の階層に応じて、各 CSR に対して最大 3 通の証明書（証明書チェーン）を取得できます。このことは、NDB スイッチごとに、CA から 3 通の証明書（root、中間、ドメイン）を取得することを意味します。各タイプの証明書を識別するには、CA に確認する必要があります。証明書の命名規則は、認定機関ごとに異なる場合があります。たとえば、test-root-ca-2048.cer（ルート）、test-ssl-ca.cer（中間）、N9K-117.cisco.com.cer（ドメイン）のようになります。

証明書はほとんどの場合、.PEM ファイル形式で共有されます。

cert.reqファイルのデータは、サードパーティの証明機関に送付する必要があります。関連する手順に従って、3 通の（証明書）ファイルを取得します。

ステップ 5 **cat** コマンドを使用して、3 通の証明書ファイルから 1 通の証明書ファイルを作成します。この連結は、ドメイン証明書、root 証明書、中間証明書の順番で行われます。**cat** コマンドの構文は、 *cat domain certificateroot certificateintermediate certificate* > *server.cer* のようになります。

例：

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

ステップ 6 新しく作成した server.cer ファイルを編集して、連結された END 行と BEGIN 行を分離します。ファイルから何も削除しないでください。

例：

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


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

（注） この手順をすべてのスイッチで繰り返します。

ステップ 7 NDB スイッチにログインします。

ステップ 8 copy コマンドを使用して、秘密キー（cert.key）と証明書を CA（server.cer）からスイッチにコピーします。

例：

```
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)...
```

（注） すべてのスイッチに対してこの手順を繰り返します。

ステップ 9 スイッチで証明書ファイル sw1-ca.pem とキーファイル sw1-ca.key を構成します。**nxapi** コマンドを使用します。

例：

```
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)#
```

（注） 複数のスイッチがある場合は、各スイッチで対応する証明書と秘密キーを構成します。

ステップ 10 **nxapi certificate** コマンドを使用して、スイッチの自己署名証明書を有効にします。

例：

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

（注） スイッチで自己署名証明書を有効にするときにエラーが生じないことを確認します。

NDB コントローラの TLS キーストアとトラストストア ファイルの作成

NDB は証明書とキーを使用して、スイッチ間の通信を保護します。キーと証明書はキーストアに補完されます。これらのファイルは、NDB に tlsTruststore および tlsKeystore ファイルとして保存されます。NDB コントローラの Java tlsKeyStore および tlsTrustStore ファイルを生成するには、次の手順を実行します。

ステップ 1 TLS ディレクトリを作成し、それに移動します。

例：

```
mkdir -p TLS
cd TLS
```

ステップ 2 *mypersonalca* の下に 3 つのディレクトリと 2 つの前提となるファイルを作成します。

例：

```
mkdir -p mypersonalca/certs
mkdir -p mypersonalca/private
mkdir -p mypersonalca/crl
echo "01" > mypersonalca/serial
touch mypersonalca/index.txt
```

コマンドを使用して、NDB に接続されている各スイッチの TLS 秘密キーと認証局（CA）ファイルを生成します。

ステップ 3 **openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out mypersonalca/certs/sw1-ca.pem -outform PEM -keyout mypersonalca/private/sw1-ca.key** コマンドを使用して、NDB に接続された各スイッチの TLS 秘密キーと認証局（CA）ファイルを生成します。

この手順により、2048 ビットのキー長の PEM 形式の TLS 秘密キーと CA ファイル（mypersonalca/certs/sw1-ca.pem、mypersonalca/private/sw1-ca.key）が生成されます。複数のスイッチがある場合は、これらのスイッチの CSR を生成するときに提供された正確な値を使用して、すべてのスイッチに対して sw1-ca.pem および sw1-ca.key ファイルを作成する必要があります。

（注） 認証局からの証明書の取得セクションで cert.key を生成するときに指定したのと同じ入力を使用します。入力に不一致があると、新しいキーが生成されることになります。

例：

```
docker@docker-virtual-machine:~/TLS$ openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out
mypersonalca/certs/sw1-ca.pem -outform PEM -keyout mypersonalca/private/sw1-ca.key
Generating a 2048 bit RSA private key
...+++
.........................................+++
writing new private key to 'mypersonalca/private/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
```

「認証局からの証明書の取得」セクションで作成した cert.key と server.cer をカレントディレクトリ（TLS）にコピーします。単一のスイッチ用の証明書とキー ファイルを選択します。これらのファイルは、コントローラに接続するすべてのスイッチに対して以前に生成されたものです。現在のスイッチの server.cer と cert.key を使用して、TLS キーストア ファイルを作成します。

ステップ 4 認証局からの証明書の取得セクションで作成した cert.key と server.cer を現在のディレクトリ（TLS）にコピーします。単一のスイッチ用の証明書とキー ファイルを選択します。これらのファイルは、コントローラに接続するすべてのスイッチに対して以前に生成されたものです。現在のスイッチの server.cer と cert.key を使用して、TLS キーストア ファイルを作成します。

複数のスイッチが接続されている場合、各スイッチに対して個別にこの手順を繰り返します。

ステップ 5 **copy** コマンドを使用して、server.cer および cert.key ファイルをコピーし、.PEM 形式に変換します。

例：

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

ステップ 6 **cat** コマンドを使用して、秘密キー（sw1-ndb-privatekey.pem）と証明書ファイル（sw1-ndb-cert.pem）を単一の .PEM ファイルに連結します。

例：

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

ステップ 7 **openssl** コマンドを使用して、.PEM ファイルを .P12 形式に変換します。指示メッセージが表示されたらエクスポートパスワードを入力します。パスワードには少なくとも 6 文字が含まれなければなりません。例：cisco123 sw1-ndb.pem ファイルは、パスワードで保護された sw1-ndb.p12 ファイルに変換されます。

例：

```
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)
```

ステップ 8 **keytool** コマンドを使用して、sw1-ndb.p12 をパスワード保護された Java キーストア（tlsKeyStore）ファイルに変換します。このコマンドは、sw1-ndb.p12 ファイルをパスワードで保護された tlsKeyStore ファイルに変換します。

例：

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

（注） デフォルトでは、「1」という名前のエイリアスが最初のスイッチに対して設定され、tlsKeyStore に保存されます。NDB コントローラが複数のスイッチを管理している場合は、すべてのスイッチに対してこの手順を繰り返します。2番目のスイッチを追加するときには、ユーティリティを使用して最初のスイッチのエイリアスの名前を変更できます。新しいスイッチのエイリアスの名前を変更するためのプロビジョニングも提供されます。たとえば、以下を参照してください。

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

ステップ 9 keytool コマンドを使用して、java tlsKeyStore のコンテンツをリスト表示して検証します。

例：

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

ステップ 10 **keytool** コマンドを使用して、証明書ファイル（sw1-ndb-cert.pem）を Java TrustStore（tlsTrustStore）ファイルに変換します。パスワードで保護された Java TrustStore（tlsTrustStore）ファイルを作成するためにパスワードを求められたら、入力します。パスワードは 6 文字以上にする必要があります。

例：

```
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
```

（注） NDB コントローラが複数のスイッチを管理している場合は、すべてのスイッチに対してこの手順を繰り返して、すべてのスイッチ キーを同じ TrustStore に追加します。次に例を示します。

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

ステップ 11 keytool コマンドを使用して、同じ tlsTrustStore 内の複数のスイッチのキーを一覧表示して確認します。

例：

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

TLS を使用した NDB の開始

TLS を使用して NDB を開始するには、次の手順を実行します。

ステップ 1 NDB サーバーにログインします。

ステップ 2 **runndb.sh** コマンドを使用して、NDB アプリケーションを停止します（実行中の場合）。

例：

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

ステップ 3 作成した tlsKeystore および tlsTruststore ファイルを NDB の構成フォルダ（ndb/configuration）にコピーします。

例：

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

ステップ 4 **runndb.sh** スクリプトを使用して、TLS で NDB アプリケーションを開始します。

例：

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

例：

デフォルトのユーザー名（admin）とデフォルト以外のパスワード（たとえば、pwd123）で NDB を起動するには：

```
./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)
```

（注） TLS を無効にするには、**./runndb.sh -notls** コマンドを実行します。TLS を無効にして NDB を開始するには、**./runndb.sh -notls -start** コマンドを実行します。TLS を無効にする前に、必ず NDB を停止してください。TLS を無効にした後、NDB サーバーに接続されているデバイスのポート番号を 80 に変更する必要があります。

Nexus Dashboard Data Broker での TLS KeyStore と TrustStore パスワードの構成

Nexus Dashboard Data Broker がパスワードで保護された TLS KeyStore と TrustStore のファイルを読み取れるようにするには、TLS KeyStore と TrustStore のパスワードを構成する必要があります。Nexus Dashboard Data Broker で TLS KeyStore と TrustStore のパスワードを構成するには、次の手順を実行します。

ステップ 1 Nexus Dashboard Data Broker サーバーにログインします。

ステップ 2 bin ディレクトリに移動します。

例：

```
cd ndb/bin
```

ステップ 3 **ndb config-keystore-passwords** コマンドを使用して、TLS KeyStore と TrustStore のパスワードを構成します。

例：

```
./ndb config-keystore-passwords --user admin --password admin --url https://ip-address_localhost:8443
 --verbose --prompt --keystore-password keystore_password --truststore-password truststore_password
```

Nexus Dashboard Data Broker が AAA（Tacacs/LDAP/Radius）で構成されており、上記のコマンドで **ndb config-keystore-passwords** が失敗し、*401 unauthorized* エラーが表示された場合：

1. `ndb` または `xnc` ディレクトリに移動します。
2. `./runndb.sh -stop` を使用して、Nexus Dashboard Data Broker サーバーを停止します。
3. フラグ `enable.LocalUser.Authentication` を *false* から *true* に変更して、有効にします。このフラグは Nexus Dashboard Data Broker の **config.ini** ファイルにあります。
4. `./runndb.sh -start` を使用して、Nexus Dashboard Data Broker サーバーを起動します。
5. **ndb config-keystore-passwords** コマンドを再度実行します。

（注） HA 環境では、クラスタ内のすべての Nexus Dashboard Data Broker サーバーに対して上記の手順を実行する必要があります。

Nexus Dashboard Data Broker で TLS を有効にすると、Nexus Dashboard Data Broker サーバーと Nexus Dashboard Data Broker スイッチ間のすべての接続がポート 443 を使用して確立されます。ポート 443 を使用するように Nexus Dashboard Data Broker のデバイス接続を変更してください。

これらの手順を正常に完了すると、ポート 443 を使用してコントローラに Nexus スイッチを追加できます。スイッチの FQDN を使用して、デバイスを Nexus Dashboard Data Broker コントローラに追加します。

スイッチの WebUI Sandbox を使用して証明書情報を確認できます。

WebUI ブラウザと NDB サーバーの間で TLS 自己署名証明書を生成する

自己署名証明書を使用して、集中モードで実行されている Web ブラウザと NDB サーバー間の通信を保護できます。このセクションでは、WebUI ブラウザと NDB アプリケーション間の通信を保護するための自己署名証明書を生成する方法について説明します。デフォルトでは、Cisco NDB は、発行先が Cisco NDB、発行元も Cisco NDB で、 デフォルトの有効性を持つデフォルトの証明書とともに出荷されます。構成フォルダの下にある **generateWebUIcertificate.sh** スクリプトを使用して、自己署名証明書を作成できます。Cisco NDB リリース 3.5 以前の場合、これらの証明書は 6 か月間有効です。Cisco NDB リリース 3.6 以降、証明書のデフォルトの有効期間は 6 か月ですが、証明書の有効期間を設定できるようになりました。

（注） NDB の自己署名 TLS 証明書は、集中化モードでのみ作成できます。

- WebUI ブラウザと集中化モードで実行されている NDB サーバーの間で TLS 自己署名証明書を生成する

WebUI ブラウザと集中型環境で実行されている NDB サーバーの間で TLS 自己署名証明書を生成する

次の手順を実行して、WebUI ブラウザと集中モードで実行されている NDB サーバーの間で TLS 自己署名証明書を生成します。

ステップ 1 NDB サーバーにログインし、カレント ディレクトリを `\ndb\configuration` に変更します。

例：

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

ステップ 2 **generateWebUIcertificate.sh** スクリプトを使用して、TLS 自己署名証明書を生成します。

例：

```
[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
*************
```

（注） **generateWebUIcertificate.sh** スクリプトは、NDB アプリケーションを再ロードして、ブラウザから NDB Java アプリケーションにアクセスしたときにブラウザがこの証明書の使用を開始するようにします。

ステップ 3 **keytool -list -v -keystore keystore_Name** コマンドを使用して、生成された証明書をデコードします。プロンプトが表示されたら、ストア パスワードを入力します。

例：

```
[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!
]
]

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

ステップ 4 自己署名証明書は、ブラウザと互換性のない JKS 形式で生成されます。したがって、ブラウザに証明書をインポートする前に、これらの証明書を PKCS12 形式に変換する必要があります。JKS 形式の証明書を PKCS12 形式に変換するには、次の手順を実行します。**keytool** コマンドを使用して、JKS 形式の証明書を PKCS12 形式に変換します。

（注） 変換する前に必ず元の証明書のコピーをとっておいてください。

例：

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

（注） **keytool** コマンドの入力は、UI 証明書を生成したときの入力と一致する必要があります。

（注） 結果として得られる証明書ファイル（keystore.p12）は PKSC12 形式です。

ステップ 5 この証明書をブラウザの信頼されたルート証明書ストアに追加します。証明書を信頼ルート証明書ストアストアに追加する方法については、それぞれの Web ブラウザのヘルプを参照してください。

ゲストシェル環境を使用して、組み込みモードで実行されている Web ブラウザと NDB サーバー間で TLS 自己署名証明書を生成する

ゲスト シェル環境を使用して、Web ブラウザと組み込みモードで実行されている NDB サーバーとの間で TLS 自己署名証明書を生成するには、次の手順を実行します。

ステップ 1 **guestshell** コマンドを使用してゲスト シェルに接続します。

例：

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

ステップ 2 カレント ディレクトリを \ndb\configuration に変更します。

例：

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

ステップ 3 /home/admin/ndb/configuration/generateWebUIcertificate.sh スクリプトを使用して、TLS 自己署名証明書を生成します。

例：

```
[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.
*************
```

（注） ブラウザからNDB Javaアプリケーションにアクセスするときに、ブラウザがこの証明書の使用を開始するようにするには、**guestshell reboot** コマンドを使用して guestshell を手動でリブートします。

ステップ 4 **keytool -list -v -keystore keystore_Name** コマンドを使用して、生成された証明書をデコードします。プロンプトが表示されたら、ストア パスワードを入力します。

例：

```
[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!
]
]
```

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

ステップ 5 自己署名証明書は、ブラウザと互換性のない JKS 形式で生成されます。ブラウザに証明書をインポートする前に、これらの証明書を PKCS12 形式に変換する必要があります。JKS 形式の証明書を PKCS12 形式に変換するには、次の手順を実行します。**keytool** コマンドを使用して、JKS 形式の証明書を PKCS12 形式に変換します。

（注） 変換する前に必ず元の証明書のコピーをとっておいてください。

例：

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

（注） **keytool** コマンドの入力は、UI 証明書を生成したときの入力と一致する必要があります。

（注） 結果として得られる証明書ファイル（keystore.p12）は PKSC12 形式です。

ステップ 6 CA 証明書を Web ブラウザの信頼ルート証明書ストアにアップロードします。証明書を信頼ルート証明書ストアストアに追加する方法については、それぞれの Web ブラウザのヘルプを参照してください。証明書を Web ブラウザにアップロードするときにプロンプトが表示されたら、証明書の作成中に作成したパスワードを使用します。

ステップ 7 ゲスト シェルを再起動して、NDB を再起動します。

WebUI ブラウザと NDB サーバー間の TLS サードパーティ証明書の生成

Web ブラウザと集中モードで実行されている NDB サーバー間の通信を保護できます。このセクションでは、CA 証明書を生成し、証明書を JKS 形式に変換し、証明書を Web ブラウザにアップロードする方法について説明します。CA証明書を生成するには、証明書署名要求（CSR）を生成し、認証局（CA）に送信して検証を受ける必要があります。オープン ソースのツールを使用して CSR を生成できます。

- WebUI ブラウザと集中モードで実行されている NDB サーバーの間で TLS サード パーティ証明書を生成する

集中型モードで実行中の WebUI ブラウザと NDB サーバーの間での TLS サードパーティ証明書の生成

WebUI ブラウザと集中モードで実行されている NDB サーバーとの間で TLS サード パーティ証明書を生成するには、次の手順に従います。

ステップ 1 **openssl req** コマンドを使用して証明書署名要求（CSR）を生成します。

例：

```
[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
```

（注） ndb-server.req（CSR）ファイルが証明書発行機関（CA）に送信されます。

（注） CA が提供する証明書をブラウザにエクスポートするときは、同じ情報を使用する必要があります。CSR ファイル cert.req が CA に送信されます。

ステップ 2 CSR 要求を確認または表示するには、openssl req コマンドを使用します。

例：

```
[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
```

ステップ 3 秘密キー ndb-server.key は、パスフレーズで保護されています。証明書の秘密キーの暗号化を解除する必要があります。**openssl rsa** コマンドを使用して秘密キーの暗号化を解除します。

例：

```
[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
```

（注） ndb-server.req ファイルのデータは、サード パーティの証明機関に送信する必要があります。関連する手順に従って、証明書ファイルを取得します。

選択する CA の階層に応じて、各 CSR に対して最大 3 通の証明書（証明書チェーン）を取得できます。このことは、NDB スイッチごとに、CA から 3 通の証明書（root、中間、ドメイン）を取得することを意味します。各タイプの証明書を識別するには、CA に確認する必要があります。証明書の命名規則は、認定機関ごとに異なる場合があります。例：qvrca2.cer（root）、hydsslg2.cer（中間）、ndb-server.cisco.com-39891.cer（ドメイン）。

証明書はほとんどの場合、.PEM ファイル形式で共有されます。

ステップ 4 cat コマンドを使用して 3 つの証明書ファイルから 1 つの証明書ファイルを作成します。この連結は、ドメイン証明書、root 証明書、中間証明書の順番で行われます。cat コマンドのシンタックス：**cat** *domain certificate root certificate intermediate certificate* > *ndb-server.cer*

例：

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

ステップ 5 新しく作成した server.cer ファイルを編集して、連結された END 行と BEGIN 行を分離します。ファイルから何も削除しないでください。

例：

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


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

ステップ 6 ndb-server.cer および ndb-server.key ファイルを使用して TLS NDB サーバー キーストア ファイルを作成します。copy コマンドを使用して、ファイルをスイッチにコピーします。

例：

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

ステップ 7 cat コマンドを使用して、秘密キーと証明書ファイルを単一の .PEM ファイルに結合します。

例：

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

ステップ 8 CA は PEM 形式の証明書を提供し、証明書の拡張子は .pem です。PEM 形式の証明書を PKCS12 形式に変換する必要があります。PEM ファイルである ndb-server-ndb.pem を **openssl pkcs12** コマンドを使用し

て、.P12 ファイル形式に変更します。指示メッセージが表示されたらエクスポート パスワードを入力します。パスワードには少なくとも 6 文字が含まれなければなりません。例：cisco123 ndb-server-ndb.pem ファイルはパスワード保護された ndb-server-ndb.p12 ファイルに変換されます。

例：

```
[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
```

ステップ 9 **keytool** コマンド を使用して、ndb-server-ndb.p12 をパスワード保護された Java キーストア（ndb-server-keystore）ファイルに変換します。このコマンドは、sw1-ndb.p12 ファイルをパスワードで保護された ndb-server-keystore ファイルに変換します。デスティネーション JKS ストアの新しいパスワードを作成し、プロンプトが表示されたらソース キーストアのパスワードを入力します。

例：

```
[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 ~]#
```

ステップ 10 **keytool** コマンドを使用して、java tlsKeyStore のコンテンツをリストして検証します。

例：

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

ステップ 11 証明書の生成中に設定したキー ストア パスワードを使用して、`jetty-ssl-context.xml`（ndb/configuration/etc に格納）を構成します。KeyStorePath、KeyStorePassword、TrustStorePath、TrustStorePassword を指定している行は、vi エディタを使用して編集できます。

例：

```
<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>
```

ステップ 12 NDB を再起動します。

ステップ 13 CA 証明書を Web ブラウザの信頼ルート証明書ストアにアップロードします。証明書を信頼ルート証明書ストア ストアに追加する方法については、それぞれの Web ブラウザのヘルプを参照してください。証

明書を Web ブラウザにアップロードするときにプロンプトが表示されたら、証明書の作成中に作成したパスワードを使用します。

ゲストシェル環境を使用して、組み込みモードで実行されているWebブラウザと NDB サーバー間で TLS サード パーティ証明書を生成する

ゲスト シェル環境を使用して、組み込みモードで実行されている Web ブラウザと NDB サーバーの間で TLS サード パーティ証明書を生成するには、次の手順に従います。

ステップ 1 feature コマンドを使用して、スイッチで bash-shell 機能を有効にします。

例：

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

ステップ 2 run コマンドを使用して、スイッチで bash-shell モードを開始します。

例：

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

ステップ 3 **openssl req** コマンドを使用して証明書署名要求（CSR）を生成します。プロンプトが表示されたら、必要な情報を入力します。

例：

```
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
```

（注） openssl コマンドは、秘密キー ndb-server.key と証明書署名要求ファイル ndb-server.req を作成します。ndb-server.req（CSR）ファイルが証明書発行機関（CA）に送信されます。

（注） CAが提供する証明書をブラウザにエクスポートするときは、同じ情報を使用する必要があります。CSR ファイル cert.req が CA に送信されます。

ステップ 4 コンテンツを表示したり、CSR 要求を確認したりするには、**openssl req** コマンドを使用します。

例：

```
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
```

ステップ 5 秘密キー ndb-server.key は、パスフレーズで保護されています。証明書の秘密キーの暗号化を解除する必要があります。**openssl rsa** コマンドを使用して秘密キーの暗号化を解除します。

例：

```
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
```

（注） 選択する CA の階層により、各 CSR に対して最大 3 つの証明書（証明書チェーン）を取得できます。このことは、NDB スイッチごとに、CA から 3 つの証明書（root、中間、ドメイン）を取得することを意味します。各タイプの証明書を識別するには、CA に確認する必要があります。証明書の命名規則は、認定機関ごとに異なる場合があります。例：qvrca2.cer（root）、hydsslg2.cer（中間）、ndb-server.cisco.com-39891.cer（ドメイン）。

証明書はほとんどの場合、.PEM ファイル形式で共有されます。

ステップ 6 cat コマンドを使用して 3 つの証明書ファイルから 1 つの証明書ファイルを作成します。この連結は、ドメイン証明書、root 証明書、中間証明書の順番で行われます。cat コマンドのシンタックス：**cat** *domain certificate root certificate intermediate certificate > ndb-server.cer*

例：

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

ステップ 7 新しく作成した server.cer ファイルを編集して、連結された END 行と BEGIN 行を分割します。ファイルで何も削除しないでください。

例：

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


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

ステップ 8 ndb-server.cer および ndb-server.key ファイルを使用して TLS NDB サーバー キーストア ファイルを作成します。**copy** コマンドを使用して、スイッチにファイルをコピーします。

例：

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

ステップ 9 **cat** コマンドを使用して、秘密キーと証明書ファイルを単一の .PEM ファイルに結合します。

例：

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

ステップ 10 CA は PEM 形式の証明書を提供し、証明書の拡張子は .pem です。PEM 形式の証明書を PKCS12 形式に変換する必要があります。PEM ファイルである ndb-server-ndb.pem を **openssl pkcs12** コマンドを使用して、.P12 ファイル形式に変更します。指示メッセージが表示されたらエクスポート パスワードを入力し

ます。パスワードには少なくとも 6 文字が含まれなければなりません。例：cisco123 ndb-server-ndb.pem ファイルはパスワード保護された ndb-server-ndb.p12 ファイルに変換されます。

例：

```
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
```

ステップ 11 証明書ファイルを NDB 構成フォルダーにコピーします。

例：

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

ステップ 12 **exit** コマンドを使用して、bash シェル モードを終了します。

例：

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

ステップ 13 guestshell コマンドを使用してゲスト シェルに接続します。

例：

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

ステップ 14 カレント ディレクトリを ndb/configuration に変更します。

例：

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

ステップ 15 **keytool** コマンドを使用して、ndb-server-ndb.p12 をパスワードで保護された Java KeyStore（ndb-server-keystore）ファイルに変換します。このコマンドは、ndb-server-ndb.p12 ファイルをパスワードで保護された ndb-server-keystore ファイルに変換します。デスティネーション JKS ストアの新しいパスワードを作成し、プロンプトが表示されたらソース キーストアのパスワードを入力します。

例：

```
[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
```

ステップ 16 **keytool** コマンドを使用して、java tlsKeyStore のコンテンツをリストして検証します。

例：

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

ステップ 17 証明書の生成中に作成したキー ストア パスワードを使用して、jetty-ssl-context.xml（ndb/etc に格納）を構成します。keystore および keystorepass で以下の行を編集するには、vi エディタを使用できます。

例：

```
<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>
```

ステップ 18 CA 証明書を Web ブラウザの信頼ルート証明書ストアにアップロードします。証明書を信頼ルート証明書ストア ストアに追加する方法については、それぞれの Web ブラウザのヘルプを参照してください。証明書を Web ブラウザにアップロードするときにプロンプトが表示されたら、証明書の作成中に作成したパスワードを使用します。

ステップ 19 NDB を再起動します。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。