Nexus Dashboard でのアプリとしての Nexus Dashboard Data Broker の TLS の管理

この章には、次の詳細が含まれています。

アプリにおける NXAPI に対して NDB サーバーと NDB スイッチの間での TLS 自己署名証明書の生成

このセクションでは、アプリ展開で 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。NDBswitch と NDB サーバー間の TLS 通信は、ポート 443 のみを使用します。

NDB サーバーと NXAPI の NDB スイッチの間で 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。

（注） TLS を構成した後でポート 80 を使用して、通信するためのコントローラを構成できません。

自己署名証明書とキーの生成

この手順で、自己署名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構成されていることを確認してください。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 に対して構成されます。

ステップ 1 ルート ユーザーとしてアプリ コンテナの 1 つにログインします。

ND/ APIC コンテナにログインするためには、Cisco Nexus Dashboard のコンテナへのログイン （13 ページ） または Cisco APIC のコンテナへのログイン （13 ページ） を参照してください。

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。

このコマンドは、証明書ファイル（sw1-ca.pem）と秘密キー（sw1-ca.key）を作成します。

```
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は、スイッチごとに証明書ファイルと秘密キーを生成します。

ステップ 3 **scp** コマンドを使用して、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。

例：

```
bash-4.2# scp sw1-ca.key admin@10.16.206.250:/
User Access Verification
Password:
sw1-ca.key
100% 1704   992.7KB/s   00:00
4.6KB/s 00:00

bash-4.2# scp sw1-ca.pem admin@10.16.206.250:/
```

```
User Access Verification
Password:
sw1-ca.key

100% 1704   992.7KB/s   00:00
4.6KB/s 00:00
```

（注） 複数のスイッチがある場合は、すべてのスイッチに対してこの手順を繰り返します。

ステップ 4 **cat** コマンドを使用して、sw1-ca.pem ファイルの内容を取得します。同じ内容をコピーします。

他のすべてのコンテナーに同じ名前のファイルを作成し、コピーした内容を vi エディタを使用してそのファイルに貼り付け、変更したファイルを保存します。同じ手順を実行して、`sw1-ca.key` ファイルの内容をすべてのコンテナにコピーします。

ステップ 5 証明書ファイル、sw1-ca.pem、およびキーファイル、sw1-ca.key を **nxapi** コマンドを使用してスイッチで構成します。

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

（注） 複数のスイッチがある場合、各スイッチに対して対応する証明書ファイルとプライベート キーを構成します。

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

例：

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

（注） スイッチで自己署名証明書を有効化する間にエラーがないことを確認します。

ステップ 7 アプリのコンテナーに root ユーザーとしてログインします。

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求められたら、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 をパスワードで保護された Java KeyStore (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の名前を変更でき、2番目のスイッチのエイリアスの名前を変更するプロビジョニングも提供されます。以下に示された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 のコンテンツを一覧表示して確認します。

例：

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

（注） すべてのコンテナーで手順 7 から 12 を繰り返します。

TLS トラストストア ファイルの作成

トラストストアは、1つ以上のスイッチに対して生成される自己署名証明書から作成されます。コントローラでは1つ以上のスイッチに対して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トラストストアを作成する方法について説明します。コントローラー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（たとえば、sw1-ndb-cert.pem、sw2-ndb-cert.pem）があります。

TLS トラストストア ファイルを生成するには、次の手順を使用します。

（注） すべてのアプリ コンテナーでこの手順を実行します。

ステップ 1 ルートユーザーとしてアプリ コンテナにログインします。

ND/ APIC コンテナにログインするためには、Cisco Nexus Dashboard のコンテナへのログイン （13 ページ） または Cisco APIC のコンテナへのログイン （13 ページ） を参照してください。

ステップ 2 keytool コマンドを使用して、sw1-ndb-cert.pem などの証明書ファイルを Java トラストストア（tlsTrustStore）ファイルに変換します。指示メッセージが表示されたときにパスワードを入力して、パスワード保護された Java TrustStore（tlsTrustStore）ファイルを作成します。パスワードは少なくとも 6 文字でなければなりません。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
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)
```

NDB コントローラが複数のスイッチを管理する場合、すべてのスイッチに対してこの手順を繰り返して、同じ TrustStore にすべてのスイッチを追加してください。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 の複数のスイッチに対するキーをリストし、検証します。

例：

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

TLS での Nexus Dashboard Data Broker の起動

TLS を使用して Nexus Dashboard Data Broker を起動するには、この手順を使用します。

ステップ 1 ルートユーザーとしてアプリ コンテナにログインします。

ND/ APIC コンテナにログインするためには、Cisco Nexus Dashboard のコンテナへのログイン （13 ページ） または Cisco APIC のコンテナへのログイン （13 ページ） を参照してください。

ステップ 2 作成した tlsKeystore および tlsTruststore ファイルをデータ ブローカーの構成フォルダーにコピーします。（/home/app/ndb/configuration - ND の場合および /home/app/local-data/configuration - APIC の場合）。

例：

```
for ND:
cp tlskeystore /home/app/ndb/configuration
cp tlsTrustStore /home/app/ndb/configuration
```

```
for APIC:
cp tlskeystore /home/app/local-data/configuration
cp tlsTrustStore /home/app/local-data/configuration
```

（注） すべてのアプリ コンテナで手順 1 と 2 を実行します。

ステップ 3 ホスト上のアプリ タイルからアプリを再起動します。

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 および TrustStore ファイルを読み取れるようにするには、TLS KeyStore および TrustStore 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。

（注） すべてのコンテナでこの手順を実行します。

ステップ 1 ルート ユーザーとしてアプリ コンテナにログインします。

ND/ APIC コンテナにログインするためには、Cisco Nexus Dashboard のコンテナへのログイン （13 ページ） または Cisco APIC のコンテナへのログイン （13 ページ） を参照してください。

ステップ 2 `bin` ディレクトリに移動します。

例：`cd /home/app/ndb/bin`

ステップ 3 **ndb config-keystore-passwords** コマンドを使用して、TLS キーストアとトラストストアのパスワードを構成します。

例：

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

このコマンドでパスワードの入力を求められたら、*admin* と入力します。

（注） KeyStore と TrustStore のパスワードのみがユーザー定義です。

TLS が Nexus Dashboard Data Broker で有効になった後で、Nexus Dashboard Data Broker サーバーと Nexus Dashboard Data Broker スイッチの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したら、ポート 443 を使用してコントローラにネクサス スイッチを追加できます。スイッチの FQDN を使用して、Nexus Dashboard Data Broker コントローラにデバイスを追加します。スイッチの WebUI サンドボックス分析を使用して、証明書情報を検証します。

アプリの NXAPI に対して NDB サーバーと NDB スイッチの間での 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。

認証局から証明書の取得

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、CA の署名を発行した公開キーを含む証明書とともに、あなた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、証明書発行機関への証明書署名要求（CSR）を生成できます。CA は CSR からのユーザー識別情報を使用して、公開キーで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 に対して構成されます。

ステップ 1 ルート ユーザーとしてアプリ コンテナの 1 つにログインします。

ND/ APIC コンテナにログインするためには、Cisco Nexus Dashboard のコンテナへのログイン （13 ページ） または Cisco APIC のコンテナへのログイン （13 ページ） を参照してください。

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ある場合があります。秘密キー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キーからパスフレーズを削除します。

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、各 NDB スイッチに対する CA から 3 つの証明書（root、中間、ドメイン）の取得を意味します。証明書のそれぞれのタイプを識別するためには、CA を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: test-root-ca-2048.cer（ルート）、test-ssl-ca.cer（中間）、N9K-117.cisco.com.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。cert.req ファイルのデータは、サードパーティの証明機関に提出する必要があります。関連する手順に従って、3 つの（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 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 certificateroot certificateintermediate certificate > server.cer*

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 行を分割します。ファイルで何かを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 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：

```
bash-4.2# scp server.cer admin@10.16.206.250:/
User Access Verification
Password:
sw1-ca.key
                                                                    100% 1704   992.7KB/s   00:00
4.6KB/s 00:00

bash-4.2# scp cert.key admin@10.16.206.250:/
User Access Verification
Password:
sw1-ca.key

100% 1704   992.7KB/s   00:00
4.6KB/s 00:00
```

（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 8 cat コマンドを使用して、server.cer ファイルの内容を取得します。同じ内容をコピーします。

他のすべてのコンテナに同じ名前のファイルを作成し、コピーした内容を vi エディターを使用してそのファイルに貼り付け、変更したファイルを保存します。

同じ手順を実行して、cert.key ファイルの内容をすべてのコンテナにコピーします。

ステップ 9 **nxapi** コマンドを使用して、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。

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.
```

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

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

例：

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

（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ないことを確認します。

Nexus Dashboard Data Broker Controller の TLS キーストアとトラストストア ファイル

Nexus Dashboard Data Broker 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。キーと証明書をキーストアに保管します。これらのファイルは、Nexus Dashboard Data Broker に tlsTruststore および tlsKeystore ファイルとして保存されます。

Java tlsKeyStore および tlsTrustStor ファイルを生成するには、次の手順を使用します。

ステップ 1 TLS ディレクトリを作成し、そこに移動します。

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 つの前提条件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、Nexus Dashboard Data Broker 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 「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ました。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 4 **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 5 上記の手順で生成された .pem ファイルをすべてのコンテナにコピーします。

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 をパスワードで保護された Java KeyStore（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」というエイリアスが最初のスイッチの tlsKeyStore に保存されます。Nexus Dashboard Data Broker 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り、最初のスイッチ エイリアスの名前を変更し、新しいスイッチのエイリアスを名前変更するためにプロビジョニングします。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 のコンテンツを一覧表示して確認します。

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。指示メッセージが表示されたときにパスワードを入力して、パスワード保護された

Java TrustStore（tlsTrustStore）ファイルを作成します。パスワードは少なくとも 6 文字でなければなりません。

例：

```
docker@docker-virtual-machine:~/TLS$ keytool -import -alias sw1 -file sw1-ndb-cert.pem -keystore
 tlsTrustStore
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
```

Nexus Dashboard Data Broker 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
```

（注） すべてのコンテナで手順 6 ～ 11 を実行します。

次のタスク

TLS を使用して Nexus Dashboard Data Broker を起動します。詳細な手順については、TLS での Nexus Dashboard Data Broker の起動 （5 ページ） を参照してください。

その後、TLS キーストアとトラストストアのパスワードを設定します。詳細な手順については、Nexus Dashboard Data Broker での TLS KeyStore と TrustStore パスワードの構成 （6 ページ） を参照してください。

Cisco Nexus Dashboard のコンテナへのログイン

この手順を使用して、Cisco Nexus Dashboard（ND）のコンテナにログインします。

ステップ 1 ssh と生成された一時的なルート パスワードを使用して、ルート ユーザーとして任意の ND ノードにログインします。

```
ssh root@10.16.206.50
root@10.16.206.50's password: <enter temporary root password here>
[root@ND-1 ~]#
```

ステップ 2 NDDB に関連する利用可能なポッドが一覧表示されます。

例：

```
[root@ND-1 ~]# kubectl -n cisco-ndb get pods
NAME          READY   STATUS    RESTARTS   AGE
ndbserver-0   1/1     Running   0          12d
ndbserver-1   1/1     Running   0          12d
ndbserver-2   1/1     Running   0          12d
```

ステップ 3 **kubectl -n cisco-ndb exec -it <pod-name> -- bash** コマンドを使用して、いずれかのポッドで実行されているコンテナへの bash シェルを取得します。

例：

```
[root@ND-1 ~]# kubectl -n cisco-ndb exec -it ndbserver-0 -- bash
```

Cisco APIC のコンテナへのログイン

この手順を使用して、Cisco APIC のコンテナにログインします。

ステップ 1 ssh と生成された一時的な root パスワードを使用して、root ユーザーとして APIC ノードのいずれかにログインします。

```
ssh root@10.16.206.247
```

```
Application Policy Infrastructure Controller
root@10.16.206.247's password: <enter temporary root password here>
```

ステップ 2 そのノードで実行されている NDDB コンテナが一覧表示されます。

例：

```
[root@apic1 ~]# docker ps | grep ndb
2bb5309cbbae  31d3a284752b  "/opt/bin/conit.bi..."  5 days ago  Up 5 days
ndbserver-c1cf9a4d-ab31-06d3-b8c0-b01840fb6cc9
```

ステップ 3 **docker exec -it <container-id> /bin/bash** コマンドを使用して、そのコンテナに bash シェルを取得します。

例：

```
[root@apic1 ~]# docker exec -it ndbserver-c1cf9a4d-ab31-06d3-b8c0-b01840fb6cc9 /bin/bash
```

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。