Implementing Certification Authority Interoperability

Certification authority (CA) interoperability is provided in support of the IP Security (IPSec), Secure Socket Layer (SSL), and Secure Shell (SSH) protocols. This module describes how to implement CA interoperability.

CA interoperability permits devices and CAs to communicate so that your device can obtain and use digital certificates from the CA. Although IPSec can be implemented in your network without the use of a CA, using a CA provides manageability and scalability for IPSec.

Note IPSec is not currently supported.

- Prerequisites for Implementing Certification Authority, on page 1
- Restrictions for Implementing Certification Authority, on page 2
- How to Implement CA Interoperability, on page 2
- Authenticate Certification Authority, on page 7
- Request Your Own Certificates, on page 10
- Configure Certificate Enrollment Using Cut-and-Paste, on page 11
- Public Key-Pair Generation in XR Config Mode, on page 14

Prerequisites for Implementing Certification Authority

The following prerequisites are required to implement CA interoperability:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You need to have a CA available to your network before you configure this interoperability feature. The CA must support Cisco Systems PKI protocol, the simple certificate enrollment protocol (SCEP) (formerly called certificate enrollment protocol [CEP]).

Restrictions for Implementing Certification Authority

The software does not support CA server public keys greater than 2048 bits.

How to Implement CA Interoperability

This section contains the following procedures:

Configure Hostname and IP Domain Name

This task configures NCS 1004 hostname and IP domain name.

You must configure the hostname and IP domain name of NCS 1004 if they have not already been configured. The hostname and IP domain name are required because NCS 1004 assigns a fully qualified domain name (FQDN) to the keys and certificates used by IPSec, and the FQDN is based on the hostname and IP domain name you assign to the NCS 1004 device. For example, a certificate named *ncs1k.example.com* is based on the NCS 1004 hostname of *ncs1k* and a device IP domain name of *example.com*.

Procedure

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:ios#configure
```

Enters mode.

Step 2 **hostname** *name*

Example:

```
RP/0/RP0/CPU0:ios(config)# hostname myhost
```

Configures the hostname of the NCS 1004 device.

Step 3 **domain name** *domain-name*

Example:

```
RP/0/RP0/CPU0:ios(config)# domain name mydomain.com
```

Configures the IP domain name of NCS 1004.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Generate RSA Key Pair

This task generates an RSA key pair.

Note

- RSA keys are auto-generated at the time of NCS 1004 boot up. Hence, step 1 is required to be configured only if the RSA key-pair is missing on NCS 1004 under some circumstances.

RSA key pairs are used to sign and encrypt IKE key management messages and are required before you can obtain a certificate for NCS 1004.

Procedure

Step 1 **crypto key generate rsa** [**usage keys** | **general-keys**] [*keypair-label*]

Example:

```
RP/0/RP0/CPU0:ios# crypto key generate rsa general-keys
```

Generates RSA key pairs.

- Use the **usage keys** keyword to specify special usage keys; use the **general-keys** keyword to specify general- purpose RSA keys.
- The *keypair-label* argument is the RSA key pair label that names the RSA key pairs.

To delete the RSA keys, use the no form: **no crypto key generate rsa**

Step 2 **crypto key zeroize rsa** [*keypair-label*]

You can run the **crypto key zeroize** command only in the `exec` mode

Example:

```
RP/0/RP0/CPU0:ios# crypto key zeroize rsa key1
```

(Optional) Deletes all RSAs from NCS 1004.

- Under certain circumstances, you may want to delete all RSA keys from NCS 1004. For example, if you believe the RSA keys were compromised in some way and should no longer be used, you should delete the keys.
- To remove a specific RSA key pair, use the *keypair-label* argument.

Step 3 **show crypto key mypubkey rsa**

Example:

```
RP/0/RP0/CPU0:ios# show crypto key mypubkey rsa
Fri Mar 27 14:00:20.954 IST
```

```
Key label: system-root-key
Type : RSA General purpose
Size : 2048
Created : 01:13:10 IST Thu Feb 06 2020
Data :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 00A93DE0 1E485EE3 0E7F0964 C48361D1 B6014BE7 A303D8D6 F7790E92 88E69C4B
 B97B7A9C D1B277E3 1569093C 82BD3258 7F67FB49 94860ECD 34498F1F 59B45757
 F32C8E8F 7CEE23EC C36A43D1 9F85C0D9 B96A14DD DD3BBD4C A1FB0888 EED210A7
 39D9A403 7ACE0F6E 39107226 CA621AD8 6E8102CA 9761B86F D33F2871 9DD16559
 AFCB4729 EFCEDBAF 83DF76E4 9A439844 EE3B1180 4022F575 99E11A2C E25BB23D
 9DD74C81 4E5C1345 D9E3CC79 1B98B1AA 6C06F004 22B901EC 36C099FE 10DE2622
 EB7CE618 9A555769 12D94C90 D9BEE5EA A664E7F6 4DF8D8D4 FE7EAB07 1EF4FEAB
 22D9E55F 62BA66A0 72153CEC 81F2639F B5F2B5C5 25E10364 19387C6B E8DB8990
 11020301 0001
Key label: system-enroll-key
Type : RSA General purpose
Size : 2048
Created : 01:13:16 IST Thu Feb 06 2020
Data :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 009DBC14 C83604E4 EB3D3CF8 5BA7FDDB 80F7E85B 427332D8 BBF80148 F0A9C281
 49F87D5C 0CEBA532 EBE797C5 7F174C69 0735D13A 493670CB 63B04A12 4BCA7134
 EE0031E9 047CAA1E 802030C5 6071E8C2 F8ECE002 CC3B54E7 5FD24E5C 61B7B7B0
 68FA2EFA 0B83799F 77AE4621 435D9DFF 1D713108 37B614D3 255020F9 09CD32E8
 82B07CD7 01A53896 6DD92B5D 5119597C 98D394E9 DBD1ABAF 6DE949FE 4A8BF1E7
 851EB3F4 60B1114A 1456723E 063E50C4 2D410906 BDB7590B F1D58480 F3FA911A
 6C9CD02A 58E68D04 E94C098F 0F0E81DB 76B40C55 64603499 2AC0547A D652412A
 BCBBF69F 76B351EE 9B2DF79D E490C0F6 92D1BB97 B905F33B FAB53C20 DDE2BB22
 C7020301 0001
```

(Optional) Displays the RSA public keys for NCS 1004.

Import Public Key to NCS 1004

This task imports a public key to NCS 1004.

A public key is imported to NCS 1004 to authenticate the user.

Procedure

Step 1 **crypto key import authentication rsa** [**usage keys** | **general-keys**] [*keypair-label*]

Example:

```
RP/0/RP0/CPU0:ios# crypto key import authentication rsa general-keys
```

Generates RSA key pairs.

- Use the **usage keys** keyword to specify special usage keys; use the **general-keys** keyword to specify general- purpose RSA keys.
- The *keypair-label* argument is the RSA key pair label that names the RSA key pairs.

Step 2 show crypto key mypubkey rsa

Example:

```
RP/0/RP0/CPU0:ios# show crypto key mypubkey rsa
Fri Mar 27 14:00:20.954 IST
Key label: system-root-key
Type : RSA General purpose
Size : 2048
Created : 01:13:10 IST Thu Feb 06 2020
Data :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 00A93DE0 1E485EE3 0E7F0964 C48361D1 B6014BE7 A303D8D6 F7790E92 88E69C4B
 B97B7A9C D1B277E3 1569093C 82BD3258 7F67FB49 94860ECD 34498F1F 59B45757
 F32C8E8F 7CEE23EC C36A43D1 9F85C0D9 B96A14DD DD3BBD4C A1FB0888 EED210A7
 39D9A403 7ACE0F6E 39107226 CA621AD8 6E8102CA 9761B86F D33F2871 9DD16559
 AFCB4729 EFCEDBAF 83DF76E4 9A439844 EE3B1180 4022F575 99E11A2C E25BB23D
 9DD74C81 4E5C1345 D9E3CC79 1B98B1AA 6C06F004 22B901EC 36C099FE 10DE2622
 EB7CE618 9A555769 12D94C90 D9BEE5EA A664E7F6 4DF8D8D4 FE7EAB07 1EF4FEAB
 22D9E55F 62BA66A0 72153CEC 81F2639F B5F2B5C5 25E10364 19387C6B E8DB8990
 11020301 0001
Key label: system-enroll-key
Type : RSA General purpose
Size : 2048
Created : 01:13:16 IST Thu Feb 06 2020
Data :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 009DBC14 C83604E4 EB3D3CF8 5BA7FDDB 80F7E85B 427332D8 BBF80148 F0A9C281
 49F87D5C 0CEBA532 EBE797C5 7F174C69 0735D13A 493670CB 63B04A12 4BCA7134
 EE0031E9 047CAA1E 802030C5 6071E8C2 F8ECE002 CC3B54E7 5FD24E5C 61B7B7B0
 68FA2EFA 0B83799F 77AE4621 435D9DFF 1D713108 37B614D3 255020F9 09CD32E8
 82B07CD7 01A53896 6DD92B5D 5119597C 98D394E9 DBD1ABAF 6DE949FE 4A8BF1E7
 851EB3F4 60B1114A 1456723E 063E50C4 2D410906 BDB7590B F1D58480 F3FA911A
 6C9CD02A 58E68D04 E94C098F 0F0E81DB 76B40C55 64603499 2AC0547A D652412A
 BCBBF69F 76B351EE 9B2DF79D E490C0F6 92D1BB97 B905F33B FAB53C20 DDE2BB22
 C7020301 0001
```

(Optional) Displays the RSA public keys for NCS 1004.

Declare Certification Authority and Configure Trusted Point

This task declares a CA and configures a trusted point.

Procedure

Step 1 **configure**

Example:

```
RP/0/0RP0RSP0/CPU0:ios:hostname# configure
```

Enters mode.

Step 2 **crypto ca trustpoint** *ca-name*

Example:

```
RP/0/RP0/CPU0:ios(config)# crypto ca trustpoint myca
```

Declares a CA.

- Configures a trusted point with a selected name so that your NCS 1004 can verify certificates issued to peers.
- Enters trustpoint configuration mode.

Step 3 **enrollment url** *CA-URL*

Example:

```
RP/0/RP0/CPU0:ios(config-trustp)# enrollment url http://ca.domain.com/certsrv/mscep/mscep.dll
```

Specifies the URL of the CA.

- The URL should include any nonstandard cgi-bin script location.

Step 4 **query url** *LDAP-URL*

Example:

```
RP/0/RP0/CPU0:ios(config-trustp)# query url ldap://my-ldap.domain.com
```

(Optional) Specifies the location of the LDAP server if your CA system supports the LDAP protocol.

Step 5 **enrollment retry period** *minutes*

Example:

```
RP/0/RP0/CPU0:ios(config-trustp)# enrollment retry period 2
```

(Optional) Specifies a retry period.

- After requesting a certificate, the NCS 1004 waits to receive a certificate from the CA. If the NCS 1004 does not receive a certificate within a period of time (the retry period) the NCS 1004 will send another certificate request.
- Range is from 1 to 60 minutes. Default is 1 minute.

Step 6 **enrollment retry count** *number*

Example:

```
RP/0/RP0/CPU0:ios(config-trustp)# enrollment retry count 10
```

(Optional) Specifies how many times the NCS 1004 continues to send unsuccessful certificate requests before giving up.

- The range is from 1 to 100.

Step 7 **rsakeypair** *keypair-label*

Example:

```
RP/0/RP0/CPU0:ios(config-trustp)# rsakeypair mykey
```

(Optional) Specifies a named RSA key pair generated using the **crypto key generate rsa** command for this trustpoint.

- Not setting this key pair means that the trustpoint uses the default RSA key in the current configuration.

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Authenticate Certification Authority

This task authenticates the CA to your NCS 1004 device.

The NCS 1004 device must authenticate the CA by obtaining the self-signed certificate of the CA, which contains the public key of the CA. Because the certificate of the CA is self-signed (the CA signs its own certificate), manually authenticate the public key of the CA by contacting the CA administrator to compare the fingerprint of the CA certificate.

Procedure

Step 1 **crypto ca authenticate** *ca-name*

Example:

```
RP/0/RP0/CPU0:ios#crypto ca authenticate myca
```

Authenticates the CA to your NCS 1004 device by obtaining a CA certificate, which contains the public key for the CA.

Step 2 **show crypto ca certificates**

Example:

```
RP/0/RP0/CPU0:ios#show crypto ca certificates
```

(Optional) Displays information about the CA certificate.

Multi-Tier Certificate Authority for Trustpoint Authentication

Table 1: Feature History Table

Feature Name	Release Information	Description
Multi-Tier Certificate Authority for Trustpoint Authentication	Cisco IOS XR Release 7.10.1	Apart from the root certificate authority (CA), you can now use a subordinate CA to issue certificates and authenticate your network devices. This feature is beneficial when you have an existing CA hierarchy where it is not the root CA but the subordinate CA that issues the leaf or certificates. In earlier releases, you could associate only a single CA, not a multi-tier CA, to a trustpoint. And, you could use only the root CA certificate to enroll the certificates. This feature modifies the show crypto ca certificates command to display the **Trusted Certificate Chain** field.

During terminal-based enrollment of a CA trustpoint, Cisco IOS XR network devices accepted only Root CA certificate. You might have some network topologies which use multi-tier CA hierarchy for enrollment purposes because it provides more flexibility and security. From Cisco IOS XR Release 7.10.1 and later, as part of terminal-based authentication, you can import a complete CA hierarchy (from the Root CA till the subordinate CA that issues the certificate) as part of a single authentication request. With this feature, you can provide a certificate chain including the Root CA and intermediate subordinate CAs as part of the terminal-based enrollment process. This feature is useful if you have an existing multi-tier CA hierarchy where the Root CA does not issue any certificates directly. And, if you want only subordinate CAs to issue certificates to authenticate your network devices.

You can have a maximum of 8 tiers, that is, a chain of CA with one Root CA and seven subordinate CAs, for trustpoint authentication.

How to Use Multi-Tier CA for Trustpoint Authentication

Use the **crypto ca authenticate** command to use multi-tier CAs for trustpoint authentication or enrollment. You must use only privacy enhanced mail (PEM)-encoded certificates for trustpoint authentication using multi-tier CAs.

The enrollment process remains the same as that of the enrollment using single-tier CA, except that you get a message on NCS 1004 console that prompts to use only PEM-encoded certificates.

Prerequisite

You must generate a key pair, import a public key and configure a trustpoint on NCS 1004 as detailed in the previous sections.

Configuration Example

```
RP/0/RP0/CPU0:ios#crypto ca authenticate test-ca
Mon Feb  6 08:17:48.943 UTC


Enter the base 64/PEM encoded certificate/certificates.
Please note: for multiple certificates use only PEM
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIF5TCCA82gAwIBAgICEAEwDQYJKoZIhvcNAQELBQAwXTELMAkGA1UEBhMCSU4x
CzAJBgNVBAgMAktBMQwwCgYDVQQHDANCR0wxDTALBgNVBAoMBENTQ08xDTALBgNV
.
.
.
/4UzeeX6ll0gGJVbDwGeIZTH00artqxHquKQ2P7eXQ1pg0PRNRqWN90SvT5yE33N
eHgbtvdHg1K6K6IAj/NGnd7xUrA1TQ4bdmouCNkgbXM/G9DwgkOOvZ8KYRP9JW57
LYIv2ZcRS2vdnZRD9JPGVig2EgcfVPtj+Q==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIF9TCCA92gAwIBAgIUD6AGesleqedhorkrJ9HWjz1RQzswDQYJKoZIhvcNAQEL
BQAwXTELMAkGA1UEBhMCSU4xCzAJBgNVBAgMAktBMQwwCgYDVQQHDANCR0wxDTAL
.
.
.
+6rMWd6BmfSy2PT3Qz5AjO2+3N1dd67qRRrX7skklkX4JXY42n5/l9PQtSp0wTBh
uy5yUAagynu0z07GczE7E9V+tJHRmNTbnd8pxLk4lTwqtiCIXwQLZA75SkwCS5wh
fn7OrV7uFjMaggNkvj0kSSOkWxqJ+j/KqMAA2zQMUV+qdvT6i+ZV44U=
-----END CERTIFICATE-----
Serial Number  : 10:01
  Subject:
 CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
 CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 12:31:40 UTC Sun Jun 14 2020
  Validity End   : 12:31:40 UTC Wed Jun 12 2030

  CRL Distribution Point
 http://10.105.236.78/crl_akshath_two_level_ca/crl.der
  SHA1 Fingerprint:
  D8E0C11ECED96F67FDBC800DB6A126676A76BD62
  Serial Number  : 0F:A0:06:7A:C9:5E:A9:E7:61:A2:B9:2B:27:D1:D6:8F:3D:51:43:3B
  Subject:
 CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
 CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 13:12:32 UTC Sun Jun 07 2020
  Validity End   : 13:12:32 UTC Sat Jun 02 2040

  CRL Distribution Point
 http://10.105.236.78/crl_akshath_two_level_ca/crl.der
  SHA1 Fingerprint:
  08E71248FB7578614442E713AC87C461D173952F

CA Certificate validated using issuer certificate.
RP/0/RP0/CPU0:ios#
```

Verification

Use the **show crypto ca certificates test1** to view the CA certificate chain. The command output displays the **Trusted Certificate Chain** field if there is one or more subordinate CAs involved in the hierarchy.

```
RP/0/RP0/CPU0:ios#show crypto ca certificates test-ca
Mon Feb  6 09:03:53.019 UTC

Trustpoint       : test-ca
==================================================
CA certificate
  Serial Number  : 10:01
  Subject:
           CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
           CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 12:31:40 UTC Sun Jun 14 2020
  Validity End   : 12:31:40 UTC Wed Jun 12 2030

  CRL Distribution Point
           http://10.105.236.78/crl_akshath_two_level_ca/crl.der
  SHA1 Fingerprint:
            D8E0C11ECED96F67FDBC800DB6A126676A76BD62
Trusted Certificate Chain
  Serial Number  : 0F:A0:06:7A:C9:5E:A9:E7:61:A2:B9:2B:27:D1:D6:8F:3D:51:43:3B
  Subject:
           CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
           CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 13:12:32 UTC Sun Jun 07 2020
  Validity End   : 13:12:32 UTC Sat Jun 02 2040

  CRL Distribution Point
           http://10.105.236.78/crl_akshath_two_level_ca/crl.der
  SHA1 Fingerprint:
            08E71248FB7578614442E713AC87C461D173952F
certificate
  Key usage      : General Purpose
  Status         : Available
  Serial Number  : 28:E5
  Subject:
           CN=test
  Issued By      :
           CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 08:49:54 UTC Mon Feb 06 2023
  Validity End   : 08:49:54 UTC Wed Mar 08 2023
  SHA1 Fingerprint:
            6C8644FA67D9CEBC7C5665C35838265F578835AB
Associated Trustpoint: test-ca
```

Request Your Own Certificates

This task requests certificates from the CA.

You must obtain a signed certificate from the CA for each of your NCS 1004 device's RSA key pairs. If you generated general-purpose RSA keys, your NCS 1004 device has only one RSA key pair and needs only one certificate. If you previously generated special usage RSA keys, your NCS 1004 device has two RSA key pairs and needs two certificates.

Procedure

Step 1 **crypto ca enroll ca-name**

Example:

```
RP/0/RP0/CPU0:ios# crypto ca enroll myca
```

Requests certificates for all of your RSA key pairs.

- This command causes your NCS 1004 to request as many certificates as there are RSA key pairs, so you need only perform this command once, even if you have special usage RSA key pairs.
- This command requires you to create a challenge password that is not saved with the configuration. This password is required if your certificate needs to be revoked, so you must remember this password.
- A certificate may be issued immediately or the NCS 1004 sends a certificate request every minute until the enrollment retry period is reached and a timeout occurs. If a timeout occurs, contact your system administrator to get your request approved, and then enter this command again.

Step 2 **show crypto ca certificates**

Example:

```
RP/0/RP0/CPU0:ios# show crypto ca certificates
```

(Optional) Displays information about the CA certificate.

Configure Certificate Enrollment Using Cut-and-Paste

This task declares the trustpoint certification authority (CA) that your NCS 1004 should use and configures that trustpoint CA for manual enrollment by using cut-and-paste.

Procedure

Step 1 **configure**

Example:

```
RP/0/0RP0RSP0/CPU0:ios:hostname# configure
```

Enters mode.

Step 2 **crypto ca trustpoint** *ca-name*

Example:

```
RP/0/RP0/CPU0:ios#crypto ca trustpoint myca
```

Declares the CA that your NCS 1004 should use and enters trustpoint configuration mode.

- Use the *ca-name* argument to specify the name of the CA.

Step 3 enrollment terminal

Example:

```
RP/0/RP0/CPU0:ios(config-trustp)# enrollment terminal
```

Specifies manual cut-and-paste certificate enrollment.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 5 **crypto ca authenticate** *ca-name*

Example:

```
RP/0/RP0/CPU0:ios# crypto ca authenticate myca
```

Authenticates the CA by obtaining the certificate of the CA.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in step 2.

Step 6 **crypto ca enroll** *ca-name*

Example:

```
RP/0/RP0/CPU0:ios# crypto ca enroll myca
```

Obtains the certificates for your NCS 1004 from the CA.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in Step 2.

Step 7 **crypto ca import** ca- *name* **certificate**

Example:

```
RP/0/RP0/CPU0:ios# crypto ca import myca certificate
```

Imports a certificate manually at the terminal.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered in Step 2.

Note
You must enter the **crypto ca import** command twice if usage keys (signature and encryption keys) are used. The first time the command is entered, one of the certificates is pasted into the NCS 1004; the second time the command is entered, the other certificate is pasted into the NCS 1004. (It does not matter which certificate is pasted first.

Step 8 show crypto ca certificates

Example:

```
RP/0/RP0/CPU0:ios# show crypto ca certificates
```

Displays information about your certificate and the CA certificate.

The following example shows how to configure CA interoperability.

Comments are included within the configuration to explain various commands.

```
RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#hostname mydevice
RP/0/RP0/CPU0:ios(config)#domain name mydomain.com
RP/0/RP0/CPU0:ios(config)#end

Uncommitted changes found, commit them? [yes]:yes

RP/0/RP0/CPU0:ios(config)#crypto key generate rsa mykey

The name for the keys will be:mykey
Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose
Keypair
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus [1024]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]

RP/0/RP0/CPU0:ios#show crypto key mypubkey rsa

Key label:mykey
Type     :RSA General purpose
Size     :1024
Created  :17:33:23 UTC Thu Sep 18 2003
Data     :
 30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00CB8D86
 BF6707AA FD7E4F08 A1F70080 B9E6016B 8128004C B477817B BCF35106 BC60B06E
 07A417FD 7979D262 B35465A6 1D3B70D1 36ACAFBD 7F91D5A0 CFB0EE91 B9D52C69
 7CAF89ED F66A6A58 89EEF776 A03916CB 3663FB17 B7DBEBF8 1C54AF7F 293F3004
 C15B08A8 C6965F1E 289DD724 BD40AF59 E90E44D5 7D590000 5C4BEA9D B5020301
 0001

! The following commands declare a CA and configure a trusted point.

RP/0/RP0/CPU0:ios#configure
RP/0/RP0/CPU0:ios(config)#crypto ca trustpoint myca
RP/0/RP0/CPU0:ios(config)#enrollment url http://xyz-ultra5
RP/0/RP0/CPU0:ios(config)#enrollment retry count 25
RP/0/RP0/CPU0:ios(config)#enrollment retry period 2
RP/0/RP0/CPU0:ios(config)#rsakeypair mykey
RP/0/RP0/CPU0:ios(config)#end

Uncommitted changes found, commit them? [yes]:yes

! The following command authenticates the CA to your device.

RP/0/RP0/CPU0:ios(config)#crypto ca authenticate myca

Serial Number  :01
Subject Name   :
cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
Issued By      :
cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
Validity Start :07:00:00 UTC Tue Aug 19 2003
Validity End   :07:00:00 UTC Wed Aug 19 2020
Fingerprint:58 71 FB 94 55 65 D4 64 38 91 2B 00 61 E9 F8 05
Do you accept this certificate?? [yes/no]:yes

! The following command requests certificates for all of your RSA key pairs.

RP/0/RP0/CPU0:ios(config)#crypto ca enroll myca

% Start certificate enrollment ...
```

```
% Create a challenge password. You will need to verbally provide this
  password to the CA Administrator in order to revoke your certificate.
% For security reasons your password will not be saved in the configuration.
% Please make a note of it.

Password:
Re-enter Password:
    Fingerprint: 17D8B38D ED2BDF2E DF8ADBF7 A7DBE35A

! The following command displays information about your certificate and the CA certificate.

RP/0/RP0/CPU0:ios#show crypto ca certificates

Trustpoint       :myca
============================================================
CA certificate
  Serial Number  :01
  Subject Name   :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Issued By      :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Validity Start :07:00:00 UTC Tue Aug 19 2003
  Validity End   :07:00:00 UTC Wed Aug 19 2020
NCS 1004 certificate
  Key usage      :General Purpose
  Status         :Available
  Serial Number  :6E
  Subject Name   :
        unstructuredName=myncs1k.mydomain.com,o=Cisco Systems
  Issued By      :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San Jose,st=CA,c=US
  Validity Start :21:43:14 UTC Mon Sep 22 2003
  Validity End   :21:43:14 UTC Mon Sep 29 2003
  CRL Distribution Point
        ldap://coax-u10.cisco.com/CN=Root coax-u10 Certificate Manager,O=Cisco Systems
```

Public Key-Pair Generation in XR Config Mode

Public Key-Pair Generation in XR Config mode supports the following key-types and key sizes in FIPS (Federal Information Processing Standard) and non-FIPS modes.

Table 2: Supported Key-Types for non-FIPS and FIPS mode

Keys-Types	Non-FIPS mode	FIPS mode
RSA	Supported for all key sizes from 512 - 4096	Supported for key sizes 2048, 3072, 4096
DSA	Supported for key sizes 512, 768, 1024	Supported for key size 2048
ECDSA	Supported for key sizes nistp256, nistp384,nistp512	Supported for key sizes nistp256, nistp384,nistp512
ED25519	Supported	Not Supported

Guidelines and Restrictions:

The following guidelines and restrictions apply for generating crypto keys-pairs in XR Config mode:

- This feature doesn't support generation of generation of **system-root-key** and **system-enroll-key**.
- The key-pairs generated in XR Config mode overwrites any previously generated key-pairs in XR EXEC mode.
- NCS 1004 doesn't support overwriting key-pairs generated in XR Config mode from XR EXEC mode.
- When you execute **no** form of the **crypto key generate** commands in XR Config Mode, it deletes only those keys generated in XR Config mode.
- NCS 1004 doesn't support deleting key-pairs generated in XR Config mode from XR EXEC mode.
- When you execute the **crypto key generate** commands in XR EXEC mode, it doesn't overwrite or delete keys generated in XR Config mode.
- The show command **show crypto key mypubkey** displays the keys generated in XR EXEC mode first, followed by the keys generated in XR Config mode.

Configuration Examples:

The following examples show the creation of key-pairs in XR Config mode:

```
RP/0/RP0/CPU0:ios# conf t
RP/0/RP0/CPU0:ios(config)#crypto key generate dsa 512
RP/0/RP0/CPU0:ios(config)#crypto key generate rsa user1 general-keys 2048
RP/0/RP0/CPU0:ios(config)#crypto key generate rsa user2 usage-keys 2048
RP/0/RP0/CPU0:ios(config)#crypto key generate rsa 2048
RP/0/RP0/CPU0:ios(config)#crypto key generate ecdsa nistp256
RP/0/RP0/CPU0:ios(config)#crypto key generate ecdsa nistp384
RP/0/RP0/CPU0:ios(config)#crypto key generate ecdsa nistp521
RP/0/RP0/CPU0:ios(config)#crypto key generate ed25519
RP/0/RP0/CPU0:ios(config)#commit
```

Use **no** form of the command in XR Config mode to delete any of the key-pairs.

System Logs and Error Messages:

NCS 1004 generates these system logs on successful creation of key-pairs:

```
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key DSA generated, label:the_default,
modBits:1024
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key ECDSA_NISTP256 generated,
label:the_default, modBits:256
```

NCS 1004 generates these system logs on deletion of key-pairs:

```
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key RSA zeroized, label:user1
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key DSA zeroized, label:the_default
```

NCS 1004 generates these error messages if you try to overwrite the key-pairs generated in XR Config Mode from XR EXEC mode:

```
RP/0/RP0/CPU0:ios#conf t
RP/0/RP0/CPU0:ios(config)#crypto key generate ed25519
RP/0/RP0/CPU0:ios(config)#commit
RP/0/RP0/CPU0:ios(config)#crypto key generate ed25519
```

```
Cannot execute the command : Operation not permitted
ce_cmd[68727]: %SECURITY-CEPKI-6-ERR_2 : Cannot      execute the command :   Operation not
 permitted
ce_cmd[68736]: %SECURITY-CEPKI-6-ERR : Key is added as part of config mode, key deletion
is not allowed , delete key from config mode
```

NCS 1004 generates these error messages if you try to delete key-pairs generated in XR Config Mode from XR EXEC mode:

```
RP/0/RP0/CPU0:ios#conf t
RP/0/RP0/CPU0:ios(config)#crypto key generate ed25519
RP/0/RP0/CPU0:ios(config)#commit
RP/0/RP0/CPU0:ios(config)#crypto key zeroize ed25519
Cannot execute the command : Operation not permitted
ce_cmd[68736]: %SECURITY-CEPKI-6-ERR_2 : Cannot execute the command :   Operation not
permitted
```

To View the Generated Key-Pairs:

You can view the key-pairs generated in XR Config mode, listed under **Public keys from config sysdb** in the following command output:

```
RP/0/RP0/CPU0:ios#show crypto key mypubkey ecdsa
Key label: the_default
Type     : ECDSA General Curve Nistp256
Degree   : 256
Created  : 11:49:22 IST Wed Apr 21 2021
Data     :
 04D6D132 2253ABD0 81449E3F 9D5CEA3A 1107950A 829E9090 8960FBD5 ABA039B7
 24A4E217 7EA47475 91C60AC7 013DBC2E EA8434D9 0BD5B0FC 694913AE 0098A4F5
 77

Key label: the_default
Type     : ECDSA General Curve Nistp521
Degree   : 521
Created  : 22:44:22 IST Thu Mar 18 2021
Data     :
 04017798 4369F493 8D0E57D1 1975FC46 CDC03A78 03A9F90E B38CA504 17DB9A64
 D1DEA6A6 D23E7E20 4D8D4D31 C7878BDB BF5EEE40 1978A889 70C5D703 BB033B77
 0FFD9201 366A9AC8 35E69BB3 97FF4E91 6B498510 39425971 C5E43858 83286088
 A6A7BF92 0EA2B416 BD4E81CE DCEB65F1 15CC75B5 91204E89 3339A168 2382CAB6
 40170131 8F


----------------------------
 Public keys from config sysdb:
 ----------------------------
 Key label: the_default
Type     : ECDSA General Curve Nistp384
Degree   : 384
Created  : 11:51:52 IST Wed Apr 21 2021
Data     :
 045F7C14 1A88C27E 9CED3FF1 7FEDFA03 B49575FA 7AD88370 BC9C7D7F F99C8917
 33620916 758BDEFC 7187E33A 2D3CCD33 14FF3267 9855A5E9 E3BD166C CE838462
 40742231 6198EE12 3E189F42 22A8149A 8E7B186D 88E728D4 7F47D565 53441061
 79
```