Configure CA Signed Certificates with IOS XE PKI

Contents

[Introduction](#)
[Prerequisites](#)
[Requirements](#)
[Components Used](#)
[Background Information](#)
[IOS XE PKI Configuration](#)
[crypto key generate](#)
[crypto pki trustpoint](#)
[crypto pki enroll](#)
[crypto pki authenticate](#)
[crypto pki import](#)
[Authenticating Peer CA certificates](#)
[Authenticating One or More Intermediate Certificates](#)
[Verification](#)
[Troubleshooting](#)
[Advanced IOS PKI Concepts](#)
[Importing a PKCS12 formatted certificate](#)
[Exporting PKCS12 or PEM certificates](#)
[Export RSA Keys](#)
[Import RSA Keys generated off-box](#)
[Delete RSA Keys](#)
[Frequently Asked Questions](#)
[Does deleting a trustpoint invalidate the CSR or a certificate chain granted from a given CSR?](#)
[Will generating a CSR on a trustpoint invalidate the existing certificate?](#)

Introduction

This document serves as a general guide for configuring IOS XE certificates signed by a 3rd party Certificate Authority (CA).

This document will detail how to both import a multi-level CA Signed chain as for the device to serve as an Identity (ID) certificate as well as how to import other 3rd party certificates for the purpose of certificate validation.

Prerequisites

Requirements

NTP and Clock time **MUST** be configured when utilizing IOS PKI features.

If an administrator does not configure NTP you may have problems with a certificate being generated with a future/past date/time. This skew in date or time can cause import problems and other issues down the road.

Sample NTP configuration:

```
ntp server 192.168.1.1
clock timezone EST -5
clock summer-time EDT recurring
```

Components Used

- Cisco router runningCisco IOS® XE17.11.1a

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

Note that some features detailed in this document may not be available in older IOS XE versions. Where possible care has been taken to document when a command or feature has been introduced or modified.

Always refer to the official documentation for IOS XE PKI features for a given version to understand any limitations or changes that may be relevant to your specific version:

Examples:

- IOS 15 M/T: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_conn_pki/configuration/15-mt/sec-pki-15-mt-book/sec-pki-overview.html
- IOS XE 16.12.x: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_conn_pki/configuration/xe-16-12/sec-pki-xe-16-12-book/sec-est-client-supp-pki.html
- IOS XE 17.x: https://www.cisco.com/c/en/us/td/docs/routers/ios/config/17-x/sec-vpn/b-security-vpn/m_sec-pki-overview-0.html

IOS XE PKI Configuration

At a high level an administrator must perform the following actions when working with IOS XE PKI certificates:

1. Create a key for use with a feature or service (**crypto key generate**)
2. Configure a trustpoint with various parameters and link the key. (**crypto pki trustpoint**)
3. Generate a certificate signing request (CSR) (**crypto pki enroll**)
4. Provide the CSR to a CA for signing (*not covered in this document*)
5. Authenticate the Root and/or intermediate CA certificates (**crypto pki authenticate**)
6. Import the device certificates (**crypto pki import**)
7. Optional: Authenticate peer CA certificates (**crypto pki authenticate**)

These steps are detailed in the upcoming sections grouped by the commands required for the given action.

crypto key generate

Many administrators have entered this command to enable Secure Socket Shell (SSH) on a router or as part of some configuration guide for a feature. However few have not dissected what the command actually does.

Take for example the commands below:

```
crypto key generate rsa general-keys modulus 2048 label rsaKey exportable
crypto key generate ec keysize 521 exportable label ecKey
```

Dissecting these commands into the specific parts will detail the usage:

- The first part of the command in black (crypto key generate) instructs the router that we will be creating a new key. There are other options such as crypto key export, crypto key import, or crypto key zerosize those will be detailed later.
- The next part of the command in **green** (rsa general-keys, ec) instructs the router exactly what type of key we are creating. For most purposes a Rivestâ€“Shamirâ€“Adleman (RSA) keypair consisting of a public/private key will be used but an administrator can also configure elliptic curve (EC) for usage with features such as those that require ECDSA certificates or for use with ECDHE handshakes.
- The command in **orange** defines the size of our key.
    - For RSA the modulus is the terminology and values such between 512-4096 are available options. The default modulus size varies by version but it is suggested to follow Cisco's best practice for next generation cryptography and utilize keys greater than 2048.
    - For EC the key-size command is required to specify the number of bits in the key. Options are 256, 384, or 512.
- The command in **purple** defines the label for this key. This is important because an administrator may need to define multiple keys for various purposes on the same IOS XE device. The label is used to specify the exact key for use with a given feature. Where possible always use a label to distinguish the keys in use and make assigning keys to features much easier. For example: label SSH, label CUBE, label HTTPS will create two keys for use with different services or features.
    - The default label for a key is the devices hostname.domain. Some devices may generate RSA keys on first boot. By not entering a label post-fix an administrator may be at risk of inadvertently overwriting/regenerating the wrong key
- The final command in **blue** is the exportable postfix. This command details that the key can be used with the **crypto pki export** command for export and use with other systems. An example may be to import into a peer High Availability device so a single key is used by both members of a HA pair or for use within troubleshooting tools such as Wireshark to decrypt RSA based TLS sessions. Whatever the reason it must be stated that the RSA keys can only be created as exportable from the start. If an administrator creates a non-exportable RSA key this key cannot be set as exportable without regenerating the key, which can have rippling affects to other features such as invalidating all certificates created using that key. That being said, an exportable key can be downgraded to non-exportable without regenerating the key by using the command **crypto key move rsa rsaKeyLabel non-exportable**

Configuration Examples:

<#root>

```
Router(config)#

crypto key generate rsa general-keys modulus 2048 label rsaKey exportable

The name for the keys will be: rsaKey

% The key modulus size is 2048 bits
% Generating 2048 bit RSA keys, keys will be exportable...
[OK] (elapsed time was 1 seconds)

Router(config)#

crypto key generate ec keysize 521 exportable label ecKey
```

```
The name for the keys will be: ecKey
```

Verification Examples:

<#root>

```
Router#
```

```
show crypto key mypubkey rsa rsaKey
```

```
% Key pair was generated at: 10:21:42 EDT Apr 14 2023
Key name: rsaKey
Key type: RSA KEYS      2048 bits
 Storage Device: not specified
 Usage: General Purpose Key
 Key is exportable. Redundancy enabled.
 Key Data:
  30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
  [..truncated..]
  9F020301 0001

Router#
```

```
show crypto key mypubkey ec ecKey
```

```
% Key pair was generated at: 10:03:05 EDT Apr 14 2023
Key name: ecKey
Key type: EC KEYS      p521 curve
 Storage Device: private-config
 Usage: Signature Key
 Key is exportable. Redundancy enabled.
 Key Data:
  30819B30 1006072A 8648CE3D 02010605 2B810400 23038186 000401A2 A77FCD34
  [..truncated..]
  93FAC967 96ADA79E 4A245881 B2AD2F4A 279A362D F390A20F C06D5845 06DA
```

crypto pki trustpoint

Trustpoints are a "folder-like" concept for storing and managing PKI Certificates within IOS XE. (Command Syntax)

At a high level:

1. Each IOS XE trustpoint can contain a single root or intermediate CA certificate loaded by way of the **crypto pki authenticate** command. Think of authenticated trustpoints as adding certificates which are now trusted by the device.
2. Each IOS XE Trustpoint can also import a single Identity (ID) certificate loaded by way of the **crypto pki import** command. The ID certificate is this devices certificate which is usually tied to some service or feature.
3. An administrator may use the **authenticate** and **import** command on the same trustpoint (which is required to import an ID certificate discussed later.). When using auth/import workflow the trustpoint will contain two certificates (root/intermediate + identity certificate).
4. When trustpoints are used for the purpose of storing trusted peer root/intermediate CA certificates only the **crypto pki authenticate** command is required. In this scenario a trustpoint will only contain the single certificate authenticated by the administrator.

Note: The upcoming sections for **crypto pki authenticate** and **crypto pki import** and later sections detailing auth/import examples for multi-level certificates will provide further context to these four bullets.

Trustpoints can have various commands configured. These commands may be used influences the values within a Certificate Signing Request (CSR) created by the device using the **crypto pki enroll** command on a trustpoint.

There are many different commands available for a trustpoint (far too many to detail in this document) but some more common examples are detailed in the both the example trustpoint and table below:

```
crypto pki trustpoint labTrustpoint
 enrollment terminal pem
 serial-number none
 fqdn none
 ip-address none
 subject-name cn=router.example.cisco.com
 subject-alt-name myrouter.example.cisco.com
 revocation-check none
 rsakeypair rsaKey
 hash sha256
```

Command	**Description**
crypto pki trustpoint labTrustpoint	Human readable configuration label for this trustpoint. Used to link to features or services in later commands.
enrollment terminal pem	Determines what action the **crypto pki enroll** command will take. In this example **enrollment terminal pem** indicates that the certificate signing request (CSR) will be output to the terminal in a Base64 PEM formatted text. Other options such as **enrollment selfsigned** can be used to create a self-signed certificate or **enrollment url** can be configured to define an HTTP URL and leverage the Simple Certificate Enrollment Protocol (SCEP) protocol. Both of these methods are outside the scope of this document.
serial-number none	Determines if the IOS XE devices serial will be added to the CSR. This also disables the prompt during the crypto pki enroll command.
fqdn none	Determines if the Fully Qualified Domain Name (FQDN) will be added to the CSR. This also disables the prompt during the crypto pki enroll command.

ip-address none	Determines if the IOS XE devices IP Address will be added to the CSR. This also disables the prompt during the crypto pki enroll command.
subject-name cn=router.example.cisco.com	Indicates the X500 formatted which will be added to the CSR.
subject-alt-name myrouter.example.cisco.com	Starting in IOS XE 17.9.1 a comma seperated list of Subject Alternate Name (SAN) values can be added to the CSR.
revocation-check none	Indicates how the IOS XE device should check for the validity of the certificate. Options such as Certificate Revocation List (CRL), Online Certificate Status Protocol (OCSP) can be used if they are supported by the Certificate Authority of choice. This is primarily used when the trustpoint is utilized by some other configured IOS XE feature or service. Revocation status is also checked when a certificate is authenticated with a trustpoint.
rsakeypair rsaKey	Instructs the command to utilize the RSA keypair with this specific label. For ECDSA certificates utilize the command "eckeypair ecKey" which references the EC Key's label
hash sha256	This command influences the type of hashing algorithm to use. Options are SHA1, SHA256, SHA384, SHA512

crypto pki enroll

The **crypto pki enroll** command is used to trigger the enrollment command on a given trustpoint. (Command Syntax)
For the example trustpoint previously displayed the command **crypto pki enroll labTrustpoint** will display the certificate signing request (CSR) to the terminal in Base64 PEM text format as shown in the example below.

This certificate signing request can now be saved in a text file or copy and pasted from the command line for the purpose of providing to any 3rd party CA to validate and sign.

<#root>

```
Router(config)#

crypto pki enroll labTrustpoint

% Start certificate enrollment ..

% The subject name in the certificate will include: cn=router.example.cisco.com
% The fully-qualified domain name will not be included in the certificate
Display Certificate Request to terminal? [yes/no]:
```

```
yes

Certificate Request follows:

-----BEGIN CERTIFICATE REQUEST-----
MIICrTCCAZUCAQAwIzEhMB8GA1UEAxMYcm91dGVyLmV4YW1wbGUuY2lzY28uY29t
[..truncated..]
mGvBGUpn+cDIdFcNVzn8LQk=
-----END CERTIFICATE REQUEST-----

---End - This line not part of the certificate request---
```

crypto pki authenticate

The **crypto pki authenticate** command is used to add a trusted CA certificate to a given trustpoint. Each trustpoint can be authenticated a single time. That is, a trustpoint can only contain a single CA root or intermediate certificate. Running the command a second time and adding a new cert will overwrite the first certificate.

With the command **enrollment terminal pem** configured the **crypto pki authenticate** command will prompt the router for a Base64 PEM formatted certificate to be uploaded via the CLI. (Command Syntax)

An administrator may authenticate a trustpoint in order to add the root and optional intermediate certificates in a certificate chain for the purpose of importing a device's ID Certificate later.

An administrators may also authenticate a trustpoint to add other trusted root CAs to the IOS XE device for the purpose of enabling trust relationships with peer devices during protocol handshakes with that peer device.

To illustrate further, a peer device may feature a certificate chain signed by "Root CA 1". In order for certificate validation during the protocol handshake between the IOS XE device and the peer device to be successful; an administrator can use **crypto pki authenticate** command to add the CA certificate to a trustpoint on the IOS XE device.

The main item to remember: Authenticating trustpoints using crypto pki authenticate is always for adding CA root or intermediate certificates to a trustpoint; not for adding identity certificates. Note that this concept is also applied to authenticating self-signed certificates from another peer device.

The example below shows how to authenticate a trustpoint from earlier using the **crypto pki authenticate** command:

<#root>

```
Router(config)#

crypto pki authenticate labTrustpoint

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
[..truncated..]
-----END CERTIFICATE-----

Certificate has the following attributes:
       Fingerprint MD5: C955FC74 7AABC184 D8A75DE7 3C9E7218
```

```
      Fingerprint SHA1: 3A99FF61 1E9E6C7B D0E567A9 96D882F5 2279C534

% Do you accept this certificate? [yes/no]:

yes

Trustpoint CA certificate accepted.
% Certificate successfully imported
```

crypto pki import

This command is used to import the identity (ID) certificate into a trustpoint. A single trustpoint can only contain a single ID certificate and issuing the command a second time will prompt to overwrite the previously imported certificate. (Command Syntax)

The example below shows how to import an Identity certificate into the example trustpoint from earlier using the **crypto pki import** command.

<#root>

```
Router(config)#

crypto pki import labTrustpoint certificate

Enter the base 64 encoded certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
[..truncated..]
-----END CERTIFICATE-----

% Router Certificate successfully imported
```

An administrator will get an error if attempting to import a certificate before the trustpoint has authenticated the the CA certificate used to directly sign this certificate.

<#root>

```
Router(config)#

crypto pki import labTrustpoint certificate

% You must authenticate the Certificate Authority before
 you can import the router's certificate.
```

Authenticating Peer CA certificates

Peer CA certificates are added to IOS XE using the same method of adding any CA certificate. That is, they are authenticated against a trustpoint using the **crypto pki authenticate** command.

The command below shows how to create a trustpoint and authenticate a peer third party CA certificate.

1. First create a trustpoint with some descriptive name which will hold the peer CA certificate
2. configure **enrollment terminal pem** so the crypto pki authenticate command asks for the certificate via the command-line.
3. Configure r**evocation-check none** to skip CRL/OCSP checking during the import process
4. Authenticate the trustpoint and provide the certificate
5. Repeat steps 1-4 for as required for peer CA certificates (remember only one CA certificate per trustpoint!)

<#root>

```
Router(config)#
crypto pki trustpoint PEER-ROOT
Router(ca-trustpoint)#
enrollment terminal pem
Router(ca-trustpoint)#
revocation-check none


Router(ca-trustpoint)#
crypto pki authenticate PEER-ROOT


Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
[..truncated..]
-----END CERTIFICATE-----

Certificate has the following attributes:
       Fingerprint MD5: 62D1381E 3E03D06A 912BAC4D 247EEF17
      Fingerprint SHA1: 3C97CBB4 491FC8D6 3D12B489 0C285481 64198EDB

% Do you accept this certificate? [yes/no]:
yes


Trustpoint CA certificate accepted.
% Certificate successfully imported
```

Authenticating One or More Intermediate Certificates

The previous examples detail how to generate a CSR using **crypto pki enroll**, authenticate the root CA certificate using **crypto pki authenticate** and then import the identity certificate using **crypto pki import**. However, when introducing intermediate certificates; the process differs slightly. Fear not, the same concepts and commands still apply! The difference lies in the how the trustpoints which hold the certificates are laid out.

Remember that each trustpoint can only contain a single Root or Intermediate CA certificate. So in an example where we have a CA Chain like below the one shown below, it is impossible to use the crypto pki

authenticate command to add more than one CA certificate:

<#root>

```
- Root CA
```

```
 - Intermediate CA 1
```

```
   - Identity Certificate
```

Solution:

1. Create a trustpoint which will hold the authenticated Root CA.
2. Then authenticate the intermediate certificate with the trustpoint used to create the CSR
3. Finally import the identity certificate into the final trustpoint.

Using the table below one can illustrate the certificate to command to trustpoint mapping with colors that correspond to the previous chain to assist with visualization.

Certificate Name	Trustpoint to use	Command to use
Root CA	crypto pki trustpoint **ROOT-CA**	crypto pki authenticate **ROOT-CA**
Intermediate CA 1	crypto pki trustpoint **labTrustpoint**	crypto pki authenticate **labTrustpoint**
Identity Certificate	crypto pki trustpoint **labTrustpoint**	crypto pki import **labTrustpoint** certificate

The same logic can be applied to a certificate chain with two intermediate CA certificates. Again colors are provided to help with the visualization of where the new Intermediate CA is applied to the IOS XE configuration.

<#root>

```
- Root CA
```

```
  - Intermediate CA 1
```

```
    - Intermediate CA 2
```

```
      - Identity Certificate
```

Certificate Name	Trustpoint to use	Command to use

Root CA	crypto pki trustpoint **ROOT-CA**	crypto pki authenticate **ROOT-CA**
Intermediate CA 1	crypto pki trustpoint **INTER-CA**	crypto pki authenticate **INTER-CA**
Intermediate CA 2	crypto pki trustpoint **labTrustpoint**	crypto pki authenticate **labTrustpoint**
Identity Certificate	crypto pki trustpoint **labTrustpoint**	crypto pki import **labTrustpoint** certificate

Looking closely one can notice two patterns:

1. All Root or Intermediate certificates are loaded into trustpoints using **crypto pki authenticate** (regardless of how many there are).
2. One can also notice that the final certificate before the device's identity certificate (read the one that directly signed the identity certificate) is always authenticated on the same trustpoint where the identity certificate is to be imported.
    - Similar to the error shown earlier, IOS XE will not let an administrator import a certificate without first authenticating the the CA certificate used to directly sign this certificate.

These two patterns above can be used for any number of intermediate certificates beyond two although in most deployments an administrator is likely to see more than two intermediate CAs in a certificate chain.

For completeness the following Root/Identity certificate table is also provided:

<#root>

```
- Root CA
```

```
  - Identity Certificate
```

Certificate Name	Trustpoint to use	Command to use
Root CA	crypto pki trustpoint **labTrustpoint**	crypto pki authenticate **labTrustpoint**
Identity Certificate	crypto pki trustpoint **labTrustpoint**	crypto pki import **labTrustpoint** certificate

Verification

- During the authentication or import process various sanity checks are performed by IOS XE to ensure the certificate is valid and well formed. These errors will be printed to the screen or logs (show logging) look for lines that start with "CRYPTO_PKI"

Some common examples are detailed below:

Valid Before/After checks are performed based on the configured time vs that found in the certificate

<#root>

```
004458:
```

```
Aug  9
 21:05:34.403: CRYPTO_PKI: trustpoint labTrustpoint authentication status = 0
%CRYPTO_PKI: Cert not yet valid or is expired -
    start date: 05:54:04 EDT
Aug 29
 2019
    end   date: 05:54:04 EDT Aug 28 2022
```

if revocation-check is not disabled IOS XE will perform a revocation-check via the configured method before importing the certificate

<#root>

```
003375: Aug  9 20:24:14:
%PKI-3-CRL_FETCH_FAIL: CRL fetch for trustpoint ROOT failed
003376: Aug  9 20:24:14.121:
CRYPTO_PKI: enrollment url not configured
```

To view details about trustpoint configuration, authenticated, or imported use the commands below:

```
show crypto pki trustpoints trustpoint_name
show crypto pki certificates trustpoint_name
show crypto pki certificates verbose trustpoint_name
```

Troubleshooting

When debugging import issues or other PKI problems utilize the following debugs.

```
debug crypto pki messages
debug crypto pki transactions
debug crypto pki validation
debug crypto pki api
debug crypto pki callback
!
debug ssl openssl error
debug ssl openssl msg
debug ssl openssl states
debug ssl openssl ext
```

Advanced IOS PKI Concepts

Importing a PKCS12 formatted certificate

Some CA providers may provide files back in PKCS#12 format (.pfx, .p12).

PKCS#12 is a special type of certificate format where the entire certificate chain from root certificate through identity certificate are bundled along with the rsa key-pair.

This format is very handy for importing with IOS XE and can be easily imported using the command below:

```
<#root>

Router(config)#

crypto pki import PKCS12-TP pkcs12 terminal password Cisco123


or
Router(config)#

crypto pki import PKCS12-TP pkcs12 ftp://cisco:cisco@192.168.1.1/certificate.pfx password Cisco123

% Importing pkcs12...
Address or name of remote host [192.168.1.1]?
Source filename [certificate.pfx]?
Reading file from ftp://cisco@192.168.1.1/certificate.pfx!
[OK - 2389/4096 bytes]
% You already have RSA keys named PKCS12.
% If you replace them, all router certs issued using these keys
% will be removed.
% Do you really want to replace them? [yes/no]:

yes

CRYPTO_PKI: Imported PKCS12 file successfully.
```

Exporting PKCS12 or PEM certificates

An administrator may export certificates to the terminal as Base64 plain text PEM, Base64 encrypted plain text, or PKCS12 format for import into other peer devices.

This is handy when bringing up new peer devices and an administrator needs to share a Root CA certificate which signed the devices identity certificate.

Some sample syntax is below:

```
<#root>

Router(config)#

crypto pki export labTrustpoint pem terminal


Router(config)#

crypto pki export labTrustpoint pem terminal 3des password Cisco!123

Router(config)#
```

```
crypto pki export labTrustpoint pkcs12 terminal password cisco!123
```

Export RSA Keys

It may be required to export RSA keys for import into some other device or for use in troubleshooting efforts. Assuming the key-pair was created as exportable the keys may be exported using the crypto key export command along with an encryption method (DES, 3DES, AES) and password.

Sample usage:

<#root>

```
Router(config)#
```

```
crypto key export rsa rsaKey pem terminal aes Cisco!123
```

```
% Key name: IOS-VG
   Usage: General Purpose Key
   Key data:
-----BEGIN PUBLIC KEY-----
[..truncated..]
-----END PUBLIC KEY-----

base64 len 1664-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,40E087AFF0886DA7C468D2084A0DECFB

[..truncated..]
-----END RSA PRIVATE KEY-----
```

If the key is not exportable an error will be shown.

<#root>

```
Router(config)#
```

```
crypto key export rsa kydavis.cisco.com pem terminal 3des mySecretPassword
```

```
% RSA keypair kydavis.cisco.com' is not exportable.
```

Import RSA Keys generated off-box

Some admins may perform RSA and certificate creation off-box, it is possible to import the RSA keys using the **crypto key import** command as shown below using the password.

<#root>

```
Router(config)#
```

```
crypto key import rsa rsaKey general-purpose exportable terminal mySecretPassword
```

```
% Enter PEM-formatted public General Purpose key or certificate.
% End with a blank line or "quit" on a line by itself.
-----BEGIN PUBLIC KEY-----
[..truncated..]
-----END PUBLIC KEY-----

% Enter PEM-formatted encrypted private General Purpose key.
% End with "quit" on a line by itself.
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,9E31AAD9B7463502
[..truncated..]
-----END RSA PRIVATE KEY-----
quit
% Key pair import succeeded.
```

Delete RSA Keys

Utilize the command **crypto key zeroize rsa rsaKey** to delete an RSA keypair named rsaKey.

Import Cisco Trusted CA Bundle via Trustpool

Trustpools vary slightly from a trustpoint but the core usage is the same. Where trustpoints usually contain a single CA certificate a trustpool will contain a number of trusted CAs.

Cisco publishes CA bundles at https://www.cisco.com/security/pki/

One common usage is to download the ios_core.p7b file using the command below:

<#root>

```
Router(config)#
```

```
crypto pki trustpool import clean url http://www.cisco.com/security/pki/trs/ios_core.p7b
```

```
Reading file from http://www.cisco.com/security/pki/trs/ios_core.p7b
Loading http://www.cisco.com/security/pki/trs/ios_core.p7b
% PEM files import succeeded.
Router(config)#
```

Frequently Asked Questions

Does deleting a trustpoint invalidate the CSR or a certificate chain granted from a given CSR?

No, once the CSR is generated and saved the trustpoint can be deleted and re-added without invalidating the CSR.

This is often used by Cisco Technical support to start fresh when authenticating/importing certificates has gone wrong.

As long as the administrator or support engineer do not regenerate RSA keys; the CSR or signed certificate chain can be imported can be authenticated/imported.

Important! Removing the trustpoint **WILL** delete any authenticated/imported certificates which could be more problematic assuming those certificates are currently in use by some service or feature.

Will generating a CSR on a trustpoint invalidate the existing certificate?

No, this is common when certificates are nearing expiry. An administrator can perform a **crypto pki enroll** command to create a new CSR and start the certificate signing process with a CA while the existing certificates that have been authenticated/imported remain in use. The moment an administrator replaces the certificates with **crypto pki authenticate**/**crypto pki import** is the moment the old certificates are replaced.