Configuring the SSL Services Module

This chapter describes how to configure the SSL Services Module from the Command Line Interface (CLI) of the module:

- Configuring Public Key Infrastructure, page 3-1
- Configuring SSL Proxy Services, page 3-40
- Configuring Certificate Authentication, page 3-45

Configuring Public Key Infrastructure

The SSL Services Module uses the SSL protocol to enable secure transactions of data through privacy, authentication, and data integrity. The SSL protocol relies upon a system of certificates, public keys, and private keys, known as a Public Key Infrastructure (PKI).

For a detailed description of the elements and functions of a Public Key Infrastructure, see the Cisco IOS PKI Overview section of the Cisco IOS Security Configuration Guide, Release 12.4 at this URL:


Certificates, which are similar to digital ID cards, verify the identity of the server to the clients and the clients to the server. Certificates are issued by certificate authorities (CAs, also known as trustpoints), and include the name of the entity to which the certificate was issued, the entity’s public key, and the time stamps that indicate the certificate’s expiration date. Your CA or trustpoint can be a third-party CA vendor or an internal CA, such as the Cisco IOS Certificate Server feature.

Public and private keys are the ciphers that are used to encrypt and decrypt information. The public key is shared without any restrictions, but the private key is never shared. Each public-private key pair works together; data that is encrypted with the public key can only be decrypted with the corresponding private key.

Each SSL module acts as an SSL proxy for up to 256 SSL clients and servers. You must configure a pair of keys for each client or server in order to apply for a certificate for authentication.

We recommend that the certificates be stored in NVRAM so the module does not need to query the certificate authority at startup to obtain the certificates or to automatically enroll. See the “Saving Your Configuration” section on page 3-28 for more information.
The SSL module authenticates certificates that it receives from external devices when you configure the SSL module as an SSL server and you configure the server proxy to authenticate the client certificate, or when you configure the SSL module as an SSL client. The SSL module validates the start time, end time, and the signature on the certificate received.

A valid certificate may have been revoked if the key pair has been compromised. If revocation checking is necessary, the SSL module downloads the certificate revocation list (CRL) from the certificate authority and looks up the serial number of the certificate received. See the “Certificate Revocation List” section on page 3-52 for information on CRLs.

The certificate can also be filtered by matching certain certificate attribute values with access control list (ACL) maps. Only authenticated certificates that are issued by trusted certificate authorities are accepted. See the “Certificate Security Attribute-Based Access Control” section on page 3-57 for information on ACLs.

Note

Only the certificate is authenticated, not the sender of the certificate. As part of the SSL handshake, the certificate sender is challenged for ownership of the private key that corresponds to the public key published in the certificate. If the challenge fails, the SSL handshake is aborted by the SSL module.

However, the SSL module cannot verify that the sender of the certificate is the expected end user or host of the communication session. To authenticate the end user or host, additional validation is necessary during the data phase, using a user name and password, bank account number, credit card number, or mother’s maiden name.

If the certificate sender is an SSL client, the SSL module can extract attributes from the client certificate and insert these attributes into the HTTP header during the data phase. The server system that receives these headers can further examine the subject name of the certificate and other attributes and then determine the authenticity of the end user or host. See the “HTTP Header Insertion” section on page 4-6 for information on configuring HTTP header insertion. See the “Client Certificate Authentication” section on page 3-46 for information on configuring client certificate authentication.

These sections describe how to configure the public key infrastructure (PKI):

- Configuring Keys and Certificates, page 3-3
- Verifying Certificates and Trustpoints, page 3-27
- Saving Your Configuration, page 3-28
- Backing Up Keys and Certificates, page 3-30
- Monitoring and Maintaining Keys and Certificates, page 3-30
- Assigning a Certificate to a Proxy Service, page 3-32
- Renewing a Certificate, page 3-33
- Automatic Certificate Renewal and Enrollment, page 3-36
- Enabling Key and Certificate History, page 3-37
- Caching Peer Certificates, page 3-38
- Configuring Certificate Expiration Warning, page 3-38
Configuring Keys and Certificates

You can configure keys and certificates using one of the following methods:

- If you are using Simple Certificate Enrollment Protocol (SCEP), configure the keys and certificates by doing the following:
  - Generate a key pair.
  - Declare the trustpoint.
  - Get the certificate authority certificate.
  - Send an enrollment request to a certificate authority on behalf of the SSL server.

See the “Configuring the Trustpoint Using SCEP” section on page 3-5 for details.

- If you are not using SCEP, configure the keys and certificates using the manual certificate enrollment (TFTP and cut-and-paste) feature by doing the following:
  - Generate or import a key pair.
  - Declare the trustpoint.
  - Get the certificate authority certificate and enroll the trustpoint using TFTP or cut-and-paste to create a PKCS10 file.
  - Request the SSL server certificate offline using the PKCS10 package.
  - Import the SSL server certificate using TFTP or cut-and-paste.

See the “Manual Certificate Enrollment” section on page 3-10 for details.

- If you are using an external PKI system, do the following:
  - Generate PKCS12 or PEM files.
  - Import this file to the module.

See the “Importing and Exporting Key Pairs and Certificates” section on page 3-19 for details.

An external PKI system is a server or a PKI administration system that generates key pairs and enrolls for certificates from a certificate authority or a key and certificate archival system. The Public-Key Cryptography Standards (PKCS) specifies the transfer syntax for personal identity information, including the private keys and certificates. This information is packaged into an encrypted file. To open the encrypted file, you must know a pass phrase. The encryption key is derived from the pass phrase.

**Note**

You do not need to configure a trustpoint before importing the PKCS12 or PEM files. If you import keys and certificates from PKCS12 or PEM files, the trustpoint is created automatically, if it does not already exist.

See Figure 3-1 for an overview on configuring keys and certificates.
Figure 3-1 Key and Certificate Configuration Overview

1. **Generate the PKCS12 or PEM files, including all of the CA certificates in the chain, on the external PKI system**

2. **Import the PKCS12 or PEM files to the SSL Services Module**

3. **Generate the RSA key pair on the SSL Module or import the key using a PEM file**

4. **Declare the trustpoint**

5. **Are you using an external PKI system?**
   - **Yes**: Generate the PKCS12 or PEM files, including all of the CA certificates in the chain, on the external PKI system
   - **No**: Are you using SCEP?
     - **Yes**: Get all of the CA certificates in the chain using cut-and-paste or TFTP
     - **No**: Enroll the trustpoint using cut-and-paste or TFTP to create a PKCS10 package

6. **Are you using SCEP?**
   - **Yes**: Enroll to get the SSL certificate using SCEP
   - **No**: Request the SSL server certificate offline using the PKCS10 package

7. **Import the SSL server certificate using cut-and-paste or TFTP**

8. **Verify the trustpoint information**

   **Steps for using SCEP**:
   - Enroll to get the SSL certificate using SCEP
   - Request the SSL server certificate offline using the PKCS10 package
   - Import the SSL server certificate using cut-and-paste or TFTP
Configuring the Trustpoint Using SCEP

To configure a trustpoint using SCEP, complete the following tasks:

- Generating RSA Key Pairs, page 3-5
- Declaring the Trustpoint, page 3-7
- Obtaining the Certificate Authority Certificate, page 3-8
- Requesting a Certificate, page 3-8

Generating RSA Key Pairs

The first key pair generated enables SSH on the module. If you are using SSH, configure a key pair for SSH. See the “Configuring SSH” section on page 2-4.

RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Aldeman. RSA algorithm is widely used by certificate authorities and SSL servers to generate key pairs. Each certificate authority and each SSL server has its own RSA key pair. The SSL server sends its public key to the certificate authority when enrolling for a certificate. The SSL server uses the certificate to prove its identity to clients when setting up the SSL session.

The SSL server keeps the private key in a secure storage, and sends only the public key to the certificate authority, which uses its private key to sign the certificate that contains the server’s public key and other identifying information about the server.

Each certificate authority keeps the private key secret and uses the private key to sign certificates for its subordinate certificate authorities and SSL servers. The certificate authority has a certificate that contains its public key.

The certificate authorities form a hierarchy of one or more levels. The top-level certificate authority is called the root certificate authority. The lower level certificate authorities are called intermediate or subordinate certificate authorities. The root certificate authority has a self-signed certificate, and it signs the certificate for the next level subordinate certificate authority, which in turn signs the certificate for the next lower level certificate authority, and so on. The lowest level certificate authority signs the certificate for the SSL server.

The SSL Services Module supports up to eight levels of certificate authority (one root certificate authority and up to seven subordinate certificate authorities). For an example of a three-level (3-tier) enrollment, see the “Example of Three-Tier Certificate Authority Enrollment” section on page 3-9.

These certificates form a chain with the server certificate at the bottom and the root certificate authority’s self-signed certificate at the top. Each signature is formed by using the private key of the issuing certificate authority to encrypt a hash digest of the certificate body. The signature is attached to the end of the certificate body to form the complete certificate.

When setting up an SSL session, the SSL server sends its certificate chain to the client. The client verifies the signature of each certificate up the chain by retrieving the public key from the next higher-level certificate to decrypt the signature attached to the certificate body. The decryption result is compared with the hash digest of the certificate body. Verification terminates when one of the certificate authority certificates in the chain matches one of the trusted certificate authority certificates stored in the client’s own database.
If the top-level certificate authority certificate is reached in the chain, and there is no match of trusted self-signed certificates, the client may terminate the session or prompt the user to view the certificates and determine if they can be trusted.

After the SSL client authenticates the server, it uses the public key from the server certificate to encrypt a secret and send it over to the server. The SSL server uses its private key to decrypt the secret. Both sides use the secret and two random numbers they exchanged to generate the key material required for the rest of the SSL session for data encryption, decryption, and integrity checking.

**Note**
The SSL Services Module supports only general-purpose keys.

When you generate general-purpose keys, only one pair of RSA keys is generated. Named key pairs allow you to have multiple RSA key pairs, enabling the Cisco IOS software to maintain a different key pair for each identity certificate. We recommend that you specify a name for the key pairs.

**Note**
The generated key pair resides in system memory (RAM). Key pairs will be lost on power failure or module reset. You must enter the `copy system:running-config nvram:startup-config` command to save the running configuration, as well as save the key pairs to the private configuration file in the module NVRAM.

To generate RSA key pairs, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>crypto key generate rsa general-keys label key-label [exportable] [modulus size]</code></td>
<td>Generates RSA key pairs.</td>
</tr>
</tbody>
</table>

1. The `exportable` keyword specifies that the key is allowed to be exported. You can specify that a key is exportable during key generation. Once the key is generated as either exportable or not exportable, it cannot be modified for the life of the key.

**Note**
When you generate RSA keys, you are prompted to enter a modulus length in bits. The SSL Services Module supports modulus lengths of 512, 768, 1024, 1536, and 2048 bits. Although you can specify 512 or 768, we recommend a minimum modulus length of 1024. A longer modulus takes longer to generate and takes longer to use, but it offers stronger security.

This example shows how to generate general-purpose RSA keys:

```
ssl-proxy(config)# crypto key generate rsa general-keys label kpl exportable
```

The name for the keys will be: kpl

Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose Keys. Choosing a key modulus greater than 512 may take a few minutes.

How many bits in the modulus [512]: 1024

Generating RSA keys.... [OK].
Chapter 3  Configuring the SSL Services Module

Configuring Public Key Infrastructure

Note

After you generate a key pair, you can test the SSL service by generating a self-signed certificate. To generate a self-signed certificate for testing, see the “Generating a Self-Signed Certificate” section on page C-1.

Declaring the Trustpoint

You should declare one trustpoint to be used by the module for each certificate.

To declare the trustpoint that your module uses and specify characteristics for the trustpoint, perform this task beginning in global configuration mode:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** | ssl-proxy(config)# crypto ca trustpoint trustpoint-label
| | Declares the trustpoint that your module should use. Enabling this command puts you in ca-trustpoint configuration mode. |
| **Step 2** | ssl-proxy(ca-trustpoint)# rsakeypair key-label
| | Specifies which key pair to associate with the certificate. |
| **Step 3** | ssl-proxy(ca-trustpoint)# enrollment [mode ra] [retry [period minutes] [count count]] url url
| | Specifies the enrollment parameters for your certificate authority. |
| **Step 4** | ssl-proxy(ca-trustpoint)# ip-address server_ip_addr
| | (Optional) Specifies the IP address of the proxy service which will use this certificate. |
| **Step 5** | ssl-proxy(ca-trustpoint)# crl [best-effort | optional | query host:[port]]
| | (Optional) Specifies how this trustpoint looks up a certificate revocation list when validating a certificate associated with this trustpoint. See the “Certificate Revocation List” section on page 3-52 for information on CRLs. |
| **Step 6** | ssl-proxy(ca-trustpoint)# subject-name line
| | (Optional) Configures the host name of the proxy service. |
| **Step 7** | ssl-proxy(ca-trustpoint)# password password
| | (Optional) Configures a challenge password. |
| **Step 8** | ssl-proxy(ca-trustpoint)# exit
| | Exits ca-trustpoint configuration mode. |

1. The trustpoint-label should match the key-label of the keys; however, this is not a requirement.
2. Some web browsers compare the IP address in the SSL server certificate with the IP address that might appear in the URL. If the IP addresses do not match, the browser may display a dialog box and ask the client to accept or reject this certificate.
3. For example, subject-name CN=server1.domain2.com, where server1 is the name of the SSL server that appears in the URL. The subject-name command uses the Lightweight Directory Access Protocol (LDAP) format.
4. Arguments specified in the subject name must be enclosed in quotation marks if they contain a comma. For example, O="Cisco, Inc.".
5. Some browsers compare the CN field of the subject name in the SSL server certificate with the hostname that might appear in the URL. If the names do not match, the browser may display a dialog box and ask the client to accept or reject the certificate. Also, some browsers will reject the SSL session setup and silently close the session if the CN field is not defined in the certificate.

This example shows how to declare the trustpoint PROXY1 and verify connectivity:

```sh
ssl-proxy(config)# crypto ca trustpoint PROXY1
ssl-proxy(config)# rsakeypair PROXY1
ssl-proxy(ca-trustpoint)# enrollment url http://exampleCA.cisco.com
ssl-proxy(ca-trustpoint)# ip-address 10.0.0.1
ssl-proxy(ca-trustpoint)# password password
ssl-proxy(ca-trustpoint)# crl optional
```
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# subject-name C=US; ST=California; L=San Jose; O=Cisco; OU=Lab; CN=host1.cisco.com
ssl-proxy(ca-trustpoint)# end
ssl-proxy# ping example.cisco.com

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.0.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
ssl-proxy#

### Obtaining the Certificate Authority Certificate

For each trustpoint, you must obtain a certificate that contains the public key of the certificate authority; multiple trustpoints can use the same certificate authority.

**Note**

Contact the certificate authority to obtain the correct fingerprint of the certificate and verify the fingerprint displayed on the console.

To obtain the certificate that contains the public key of the certificate authority, perform this task in global configuration mode:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy(config)# crypto ca authenticate trustpoint-label</td>
<td>Obtains the certificate that contains the public key of the certificate authority. Enter the same trustpoint_label that you entered when declaring the trustpoint.</td>
</tr>
</tbody>
</table>

This example shows how to obtain the certificate of the certificate authority:

ssl-proxy(config)# crypto ca authenticate PROXY1
Certificate has the following attributes:
Fingerprint: A8D09689 74FB6587 02BFE0DC 2200B38A
% Do you accept this certificate? [yes/no]: y
Trustpoint CA certificate accepted.
ssl-proxy(config)# end
ssl-proxy#

### Requesting a Certificate

You must obtain a signed certificate from the certificate authority for each trustpoint.

To request signed certificates from the certificate authority, perform this task in global configuration mode:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy(config)# crypto ca enroll trustpoint-label1</td>
<td>Requests a certificate for the trustpoint.</td>
</tr>
</tbody>
</table>

1. You have the option to create a challenge password that is not saved with the configuration. This password is required in the event that your certificate needs to be revoked, so you must remember this password.
Note

If your module or switch reboots after you have entered the `crypto ca enroll` command but before you have received the certificates, you must reenter the command and notify the certificate authority administrator.

This example shows how to request a certificate:

```plaintext
ssl-proxy(config)# crypto ca enroll PROXY1
  %
  % Start certificate enrollment..
  %
  % The subject name in the certificate will be: C=US; ST=California; L=San Jose; O=Cisco;
  % OU=Lab; CN=host1.cisco.com
  % The subject name in the certificate will be: host.cisco.com
  % The serial number in the certificate will be: 00000000
  % The IP address in the certificate is 10.0.0.1
  %
  % Certificate request sent to Certificate Authority
  % The certificate request fingerprint will be displayed.
  % The 'show crypto ca certificate' command will also show the fingerprint.
  Fingerprint: 470DE382 65D8156B 0F84C2AF 4538B913

ssl-proxy(config)# end
```

After you configure the trustpoint, see the “Verifying Certificates and Trustpoints” section on page 3-27 to verify the certificate and trustpoint information.

Example of Three-Tier Certificate Authority Enrollment

The SSL Services Module supports up to eight levels of certificate authority (one root certificate authority and up to seven subordinate certificate authorities).

The following example shows how to configure three levels of certificate authority:

Generating the Keys

```plaintext
ssl-proxy(config)# crypto key generate rsa general-keys label key1 exportable
  The name for the keys will be: key1
  Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.
  How many bits in the modulus [512]: 1024
  % Generating 1024 bit RSA keys ...[OK]
```

Defining the Trustpoints

```plaintext
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.1
ssl-proxy(ca-trustpoint)#
ssl-proxy(config)# exit
ssl-proxy(config)# crypto ca trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.2
ssl-proxy(ca-trustpoint)#
ssl-proxy(config)# exit
ssl-proxy(config)# crypto ca trustpoint tp-proxy1
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# password cisco
ssl-proxy(ca-trustpoint)# subject CN=ste.cisco.com
ssl-proxy(ca-trustpoint)# rsakeypair key1
```
ssl-proxy(ca-trustpoint)# show
  enrollment url tftp://10.1.1.3
  serial-number
  password 7 0205D480B09
  subject-name CN=ste.cisco.com
  rsakeypair key1
end

ssl-proxy(ca-trustpoint)# exit

Authenticating the Three Certificate Authorities (One Root And Two Subordinate Certificate Authorities)

ssl-proxy(config)# crypto ca authenticate 3tier-root
Certificate has the following attributes:
  Fingerprint:84E470A2 38176CB1 AA0476B9 C0B4F478
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.

ssl-proxy(config)#

ssl-proxy(config)# crypto ca authenticate 3tier-sub1
Certificate has the following attributes:
  Fingerprint:FE89FB0D BF8450D7 9934C926 6C66708D
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.

ssl-proxy(config)#

ssl-proxy(config)# crypto ca authenticate tp-proxy1
Certificate has the following attributes:
  Fingerprint:6E53911B E29AE44C ACE773E7 26A098C3
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.

Enrolling with the Third Level Certificate Authority

ssl-proxy(config)# crypto ca enroll tp-proxy1
% % Start certificate enrollment ..
% The fully-qualified domain name in the certificate will be:ste.
% The subject name in the certificate will be:ste.
% The serial number in the certificate will be:B0FFF0C2
% Include an IP address in the subject name? [no]: Request certificate from CA? [yes/no]:yes
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

ssl-proxy(config)# Fingerprint: 74390E57 26F89436 6FC52ABE 24E23CD9

ssl-proxy(config)#
*Apr 18 05:10:20.963:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority

Manual Certificate Enrollment

The Manual Certificate Enrollment (TFTP and Cut-and-Paste) feature allows you to generate a certificate request and accept certificate authority certificates as well as router certificates. These tasks are accomplished with a TFTP server or manual cut-and-paste operations. You may want to use TFTP or manual cut-and-paste enrollment in the following situations:

- Your certificate authority does not support Simple Certificate Enrollment Protocol (SCEP) (which is the most commonly used method for sending and receiving requests and certificates).
- A network connection between the router and certificate authority is not possible (which is how a router running Cisco IOS software obtains its certificate).
Configure the Manual Certificate Enrollment (TFTP and cut-and-paste) feature as described at this URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t13/ftmancrt.htm

**Note**

If the certificate revocation list (CRL) fails to download because the CRL server is unreachable or the CRL download path does not exist, the certificate might fail to import. You should make sure all trustpoints that are linked to the import process are able to download the CRL. If the CRL path does not exist, or if the CRL server is unreachable, then you should enter the `crl optional` command for all trustpoints that are linked to the import process. Enter the `show crypto ca certificates` command to display information for all certificates, and obtain a list of associated trustpoints from the display of the certificate authority certificate. Enter the `crl optional` command for all these trustpoints.

For example, in a three-tier certificate authority hierarchy (root CA, subordinate CA1, and subordinate CA2), when you import the subordinate CA1 certificate, enter the `crl optional` command for all the trustpoints associated with root CA. Similarly, when you import the subordinate CA2 certificate, enter the `crl optional` command for all the trustpoints associated with root CA and subordinate CA1.

After you successfully import the certificate, you can restore the original CRL options on the trustpoints.

---

**Example 1: Configuring Certificate Enrollment Using TFTP (One-Tier Certificate Authority)**

1. Configure the trustpoint:
   
   ```
   ssl-proxy# configure terminal
   Enter configuration commands, one per line. End with CNTL/Z.
   ssl-proxy(config)# crypto ca trustpoint tftp_example
   ssl-proxy(config)# enrollment url tftp://10.1.1.2/win2k
   ssl-proxy(config)# rsakeypair pair3
   ssl-proxy(config)# exit
   ```

2. Request a certificate for the trustpoint:

   ```
   ssl-proxy(config)# crypto ca enroll tftp_example
   % Start certificate enrollment ..
   % The fully-qualified domain name in the certificate will be: ssl-proxy.cisco.com
   % The subject name in the certificate will be: ssl-proxy.cisco.com
   % Include the router serial number in the subject name? [yes/no]: yes
   % The serial number in the certificate will be: 00000000
   % Include an IP address in the subject name? [no]:
   % Send Certificate Request to tftp server? [yes/no]: yes
   % Certificate request sent to TFTP Server
   % The certificate request fingerprint will be displayed.
   % The 'show crypto ca certificate' command will also show the fingerprint.
   ssl-proxy(config)# Fingerprint: D012D925 96F4B5C9 661FEC1E 207786B7
   ```
3. Obtain the certificate that contains the public key of the certificate authority:

```plaintext
ssl-proxy(config)# crypto ca auth tftp_example
Loading win2k.ca from 10.1.1.2 (via Ethernet0/0.168): !
[OK - 1436 bytes]

Certificate has the following attributes:
Fingerprint: 2732ED87 965F8FEB F89788D4 914B877D
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
ssl-proxy(config)#
```

4. Import the server certificate:

```plaintext
ssl-proxy(config)# crypto ca import tftp_example cert
% The fully-qualified domain name in the certificate will be: ssl-proxy.cisco.com
Retrieve Certificate from tftp server? [yes/no]: yes
% Request to retrieve Certificate queued
ssl-proxy(config)#

Loading win2k.crt from 10.1.1.2 (via Ethernet0/0.168): !
[OK - 2112 bytes]

*Apr 15 12:02:33.535: %CRYPTO-6-CERTRET: Certificate received from Certificate Authority
ssl-proxy(config)#
```

**Example 2: Configuring Certificate Enrollment Using Cut-and-Paste (One-Tier Certificate Authority)**

1. Generate the RSA key pair:

```plaintext
ssl-proxy(config)# crypto key generate rsa general-keys label CSR-key exportable
The name for the keys will be:CSR-key
Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose Keys. Choosing a key modulus greater than 512 may take a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys ...[OK]
```

2. Configure the trustpoints:

```plaintext
ssl-proxy(config)# crypto ca trustpoint CSR-TP
ssl-proxy(ca-trustpoint)# rsakeypair CSR-key
ssl-proxy(ca-trustpoint)# serial
ssl-proxy(ca-trustpoint)# subject-name CN=abc, OU=hss, O=cisco
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
```

3. Request a certificate for the trustpoint:

```plaintext
ssl-proxy(config)# crypto ca enroll CSR-TP
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=abc, OU=hss, O=cisco
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:B0FFF22E
% Include an IP address in the subject name? [no]: no
Display Certificate Request to terminal? [yes/no]: yes
```
Certificate Request follows:

```
MIIBwjCCASsCAQAwYTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEMMAoG
A1UEAxMDYWJjMTMwDwYDVQQFEwhCMEZGRjIyRTAgBgkqhkiG9w0BAQQQFAOByQg
RjAQMAGA1UDEwDB/wQcEGEG0DANBgkqhkiG9w0BAQQQFA0BQgQCIIXO/6/hiiXH
DAAsQOpqLoilMr8PF8zbd1pLVBSorI451l/hhP2NY9yg2gAiMTRfGoMm79Jy
G0tUuCyFPYkpboU5SVTGVHhrVhIVeke1Lq9d1kfbkRmJnHBaBzBv5DNCv11SIMX
KULG70u0aU6eGnx8Mq6rNwsoToF4WrlPg==
---End - This line not part of the certificate request---
```

Redisplay enrollment request? [yes/no]: no

4. Import the certificate authority certificate:

```
ssl-proxy(config)# crypto ca authenticate CSR-TP
```

Enter the base 64 encoded CA certificate.

End with a blank line or the word "quit" on a line by itself

```
-----BEGIN CERTIFICATE-----
MIICxzCCAjCgAwIBAgIBADANBgkqhkiG9w0BAQQFADBSMQswCQYDVQQGEwJBVTET
MBEGA1UUECBMFU2tZEs1i7dGfZTtHMBgA1UEBcMfYS50ZXJuZ21lcyBbMQAw
BkgrTH9nMjwqCQYDVQQDEwJXaXN0b3JzMQswCQYDVQQKEwNjaXNjbzEMMAoG
A1UEAxMjM4MDIaFw06MDYyMDE5MDAyMDhyMF4wDQYJKoZIhvcNAQEBBQADgY0
AMIGJAoGBALt7O6tt30lBVVK1qAE/agsuzIaa15YZft3bDb9t3pPncKho1vBtG
kpiJLPWGZPjdbtejxQ
```

Certificate has the following attributes:

```
Fingerprint:B8B35B00 095573D0 D3B8FA03 B6CA8934
```

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

Trustpoint CA certificate accepted.

% Certificate successfully imported

```
ssl-proxy(config)#
```

5. Import the server certificate (the server certificate is issued by the certificate authority whose certificate is imported in Step 4):

```
ssl-proxy(config)# crypto ca import CSR-TP certificate
```

Enter the base 64 encoded certificate.

End with a blank line or the word "quit" on a line by itself

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

Certificate has the following attributes:

```
Fingerprint:B8B35B00 095573D0 D3B8FA03 B6CA8934
```

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

Trustpoint CA certificate accepted.

% Certificate successfully imported
Example 3: Configuring Certificate Enrollment Using TFTP (Three-Tier Certificate Authority)

1. Generate the RSA key pair:

   ```
   ssl-proxy(config)# crypto key generate rsa general-keys label test-3tier exportable
   The name for the keys will be: test-3tier
   Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose Keys. Choosing a key modulus greater than 512 may take a few minutes.
   
   How many bits in the modulus [512]: 1024
   % Generating 1024 bit RSA keys ... [OK]
   ```

2. Configure the trustpoint:

   ```
   ssl-proxy(config)# crypto ca trustpoint test-3tier
   ssl-proxy(ca-trustpoint)# serial-number
   ssl-proxy(ca-trustpoint)# password cisco
   ssl-proxy(ca-trustpoint)# subject CN=test-3tier, OU=hss, O=Cisco
   ssl-proxy(ca-trustpoint)# rsakeypair test-3tier
   ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-3tier
   ssl-proxy(ca-trustpoint)# exit
   ```

3. Generate the certificate signing request (CSR) and send it to the TFTP server:

   ```
   ssl-proxy(config)# crypto ca enroll test-3tier
   %
   % Start certificate enrollment ..
   % The subject name in the certificate will be: CN=test-3tier, OU=hss, O=Cisco
   % The fully-qualified domain name in the certificate will be: ssl-proxy.cisco.com
   % The subject name in the certificate will be: ssl-proxy.cisco.com
   % The serial number in the certificate will be: B0FF22E
   % Include an IP address in the subject name? [no]:
   % Send Certificate Request to tftp server? [yes/no]: yes
   % Certificate request sent to TFTP Server
   % The certificate request fingerprint will be displayed.
   % The 'show crypto ca certificate' command will also show the fingerprint.
   
   ssl-proxy(config)# Fingerprint: 19B07392 319B2ACF 58FABE5C 52798971
   ```

4. Use the CSR to acquire the SSL certificate offline from the third-level certificate authority.
5. Authenticate the three certificate authorities (one root and two subordinate certificate authorities):

```
ssl-proxy(config)# crypto ca trustpoint test-1tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-1tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate test-1tier
Loading test-1tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1046 bytes]
Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DCC3F AAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
```

```
ssl-proxy(config)# crypto ca trustpoint test-2tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-2tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate test-2tier
Loading test-2tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1554 bytes]
Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
```

```
ssl-proxy(config)# crypto ca trustpoint test-3tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-3tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(config)# crypto ca authenticate test-3tier
Loading test-3tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1545 bytes]
Certificate has the following attributes:
Fingerprint:2F2E44AC 609644FA 5B4B6B26 FDBFE569
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
```

6. Import the server certificate:

```
ssl-proxy(config)# crypto ca import test-3tier certificate
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
Retrieve Certificate from tftp server? [yes/no]:yes
% Request to retrieve Certificate queued

ssl-proxy(config)#
Loading test-3tier.crt from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1608 bytes]

ssl-proxy(config)#
*Nov 25 21:52:36.299:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
ssl-proxy(config)# ^Z
```

Example 4: Configuring Certificate Enrollment Using Cut-and-Paste (Three-Tier Certificate Authority)

1. Generate the RSA key pair:

```
ssl-proxy(config)# crypto key generate rsa general-keys label tp-proxy1 exportable
The name for the keys will be:tp-proxy1
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```
2. Configure the trustpoint:

```bash
ssl-proxy(config)# crypto ca trustpoint tp-proxy1
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# rsakeypair tp-proxy1
ssl-proxy(ca-trustpoint)# serial
ssl-proxy(ca-trustpoint)# subject-name CN=test
ssl-proxy(ca-trustpoint)# exit
```

3. Request a certificate for the trustpoint:

```bash
ssl-proxy(config)# crypto ca enroll tp-proxy1
% Start certificate enrollment ..
% The subject name in the certificate will be: CN=test
% The fully-qualified domain name in the certificate will be: ssl-proxy.
% The subject name in the certificate will be: ssl-proxy.
% The serial number in the certificate will be: B0FFF14D
% Include an IP address in the subject name? [no]: no
Display Certificate Request to terminal? [yes/no]: yes
Certificate Request follows:

```
---End - This line not part of the certificate request---
Redisplay enrollment request? [yes/no]: no
```

4. Get the certificate request from Step 3 signed by a third-level certificate authority.

5. Define and import all certificate authorities (one root and two subordinate certificate authorities).

   a. Define two trustpoints for root certificate authority and subordinate 1 certificate authority.

   b. Import the root certificate authority certificate:

   ```bash
   ssl-proxy(config)# crypto ca authenticate 3tier-root
   Enter the base 64 encoded CA certificate.
   End with a blank line or the word "quit" on a line by itself
   ```

   ```bash
   -----BEGIN CERTIFICATE-----
   MIIC1zCCAgAgIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
   MQswCQYDVQQGEwJVUzETMBEGA1UECBMDSaWZvcm5pY2hwb3NlcyB0MS8wDQYJKoZIhvc
   ---End - This line not part of the certificate request---
   Redisplay enrollment request? [yes/no]: no
   ```

   c. Define two trustpoints for root certificate authority and subordinate 2 certificate authority.
Chapter 3  Configuring the SSL Services Module

Configuring Public Key Infrastructure

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAA B4747C10
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

c. Import subordinate 1 certificate authority certificate:

ssl-proxy(config)# crypto ca authenticate 3tier-sub1

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

-----BEGIN CERTIFICATE-----
MIIETzCCA/mgAwIBAgIKGj0cBwAAAAAADjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQEEDJvzETTBMAGA1UCEMBMY2FsaWvc5p5TERMAAGA1UEBEwXc2ltcHNvbi1k
ZXZ0ZXN0LXJvb3QtQ0GCEGnVMc1P4ve4Q5mUWCdWwXAwgZcGAA1UdHwSBjzCBj
DBDoEGgP4Y9aHR0cDovL2Npc2NvLWw4ajZvaHBucl9zaW1wc29uLWRldnRlc3Qt
cm9vdC1DQS5jcnQwDQYJKoZIhvcNAQEFBQADQQA6kAV3Jx/BOr2hlSp9ER36ZkDJ
NIW93gNt2MkpcA07RmcrHln6q5RJ9WbTxdFbn0NDgsag1BC0wn97XErH2Zow
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
% Certificate successfully imported
d. Import the subordinate 2 certificate authority certificate:

```
ssl-proxy(config)# crypto ca authenticate tp-proxy1
```

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

```
-----BEGIN CERTIFICATE-----
MIIEVzCCAqAwIBAgIkiBK/1xAAAACAAAABjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWVuY2VkJTAlBgNVBAMTBHRlc3QgQ2VydEVucm9s
ZXMwMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEBBiMDQgUHRlc3QgQ2Vyd
EVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEBBiMDQgUHRlc3
QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEBBiMDQg
UHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEB
BiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHk
YwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUH
AgEWHkYwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKw
YBBQUHAgEWHkYwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQE
BCwYIKwYBBQUHAgEWHkYwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqG
SIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8
-----END CERTIFICATE-----
```

Certificate has the following attributes:
Fingerprint:2F2E44AC 609644FA 5B4B6B26 FDBFE569
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.

e. Import the server certificate:

```
ssl-proxy(config)# crypto ca import tp-proxy1 certificate
```

```
-----BEGIN CERTIFICATE-----
MIIEVzCCAqAwIBAgIkiBK/1xAAAACAAAABjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWVuY2VkJTAlBgNVBAMTBHRlc3QgQ2VydEVucm9s
ZXMwMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEBBiMDQgUHRlc3QgQ2Vyd
EVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEBBiMDQgUHRlc3
QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEBBiMDQg
UHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEB
BiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUHAgEWHk
YwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKwYBBQUH
AgEWHkYwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQEBCwYIKw
YBBQUHAgEWHkYwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqGSIb3DQE
BCwYIKwYBBQUHAgEWHkYwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8GCSqG
SIb3DQEBCwYIKwYBBQUHAgEWHkYwYQEBBiMDQgUHRlc3QgQ2VydEVucm9sZXMhMjAxNzA8
-----END CERTIFICATE-----
```

Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
% Certificate successfully imported
Importing and Exporting Key Pairs and Certificates

You can import and export key pairs and certificates using either the PKCS12 file format or privacy-enhanced mail (PEM) file format.

To import or export key pairs and certificates, see one of these sections:

- Importing and Exporting a PKCS12 File, page 3-20
- Importing and Exporting PEM Files, page 3-21

### Note

A test PKCS12 file (test/testssl.p12) is embedded in the SSL software on the module. You can install the file into NVRAM for testing purposes and for proof of concept. After the PKCS12 file is installed, you can import it to a trustpoint and then assign it to a proxy service configured for testing. For information on installing the test PKCS12 file, see the “Importing the Embedded Test Certificate” section on page C-4.

### Note

If the certificate revocation list (CRL) fails to download because the CRL server is unreachable or the CRL download path does not exist, the certificate might fail to import. You should make sure all trustpoints that are linked to the import process are able to download the CRL. If the CRL path does not exist, or if the CRL server is unreachable, then you should enter the `crl optional` command for all trustpoints that are linked to the import process. Enter the `show crypto ca certificates` command to display information for all certificates, and obtain a list of associated trustpoints from the display of the certificate authority certificate. Enter the `crl optional` command for all these trustpoints.

For example, in a three-tier certificate authority hierarchy (root CA, subordinate CA1, and subordinate CA2), when you import the subordinate CA1 certificate, enter the `crl optional` command for all the trustpoints associated with root CA. Similarly, when you import the subordinate CA2 certificate, enter the `crl optional` command for all the trustpoints associated with root CA and subordinate CA1.

After you successfully import the certificate, you can restore the original CRL options on the trustpoints.
Importing and Exporting a PKCS12 File

You can use an external PKI system to generate a PKCS12 file and then import this file to the module.

**Note**
When creating a PKCS12 file, include the entire certificate chain, from server certificate to root certificate, and public and private keys. You can also generate a PKCS12 file from the module and export it.

**Note**
Imported key pairs cannot be exported.

**Note**
If you are using SSH, we recommend using SCP (secure file transfer) when importing or exporting a PKCS12 file. SCP authenticates the host and encrypts the transfer session.

To import or export a PKCS12 file, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy(config)# crypto ca (import</td>
<td>Imports or exports a PKCS12 file.</td>
</tr>
<tr>
<td>export) trustpoint_label pkcs12 (scp:</td>
<td>Note You do not need to configure a trustpoint before importing the PKCS12 file.</td>
</tr>
<tr>
<td>ftp:</td>
<td>Importing keys and certificates from a PKCS12 file creates the trustpoint automatically, if it does not already exist.</td>
</tr>
<tr>
<td>nvram:</td>
<td></td>
</tr>
<tr>
<td>rcp:</td>
<td></td>
</tr>
<tr>
<td>tftp:</td>
<td></td>
</tr>
<tr>
<td>[pkcs12_filename] pass_phrase²</td>
<td></td>
</tr>
</tbody>
</table>

1. If you do not specify the *pkcs12_filename* value, you will be prompted to accept the default filename (the default filename is the *trustpoint_label* value) or enter the filename. For *ftp:* or *tftp:*, include the full path in the *pkcs12_filename* value.
2. You will receive an error if you enter the pass phrase incorrectly.

This example shows how to import a PKCS12 file using SCP:

ssl-proxy(config)# crypto ca import TP2 pkcs12 scp: sky is blue
Address or name of remote host []? 10.1.1.1
Source username [ssl-proxy]? admin-1
Source filename [TP2]? /users/admin-1/pkcs12/TP2.p12
Password:password
Sending file modes:C0644 4379 TP2.p12

ssl-proxy(config)#

*Aug 22 12:3:00.531:*%CRYPTO-6-PKCS12IMPORT_SUCCESS:*PKCS #12 Successfully Imported.

ssl-proxy(config)#
This example shows how to export a PKCS12 file using SCP:

```
ssl-proxy(config)# crypto ca export TP1 pkcs12 scp: sky is blue
Address or name of remote host []? 10.1.1.1
Destination username [ssl-proxy]? admin-1
Destination filename [TP1]? TP1.p12

Password:

Writing TP1.p12 Writing pkcs12 file to scp://admin-1@10.1.1.1/TP1.p12

Password: !
CRYPTO_PKI: Exported PKCS12 file successfully.
```

This example shows how to import a PKCS12 file using FTP:

```
ssl-proxy(config)# crypto ca import TP2 pkcs12 ftp: sky is blue
Address or name of remote host []? 10.1.1.1
Source filename [TP2]? /admin-1/pkcs12/PK-1024
Loading /admin-1/pkcs12/PK-1024 !
[OK - 4339/4096 bytes]
```

This example shows how to export a PKCS12 file using FTP:

```
ssl-proxy(config)# crypto ca export TP1 pkcs12 ftp: sky is blue
Address or name of remote host []? 10.1.1.1
Destination filename [TP1]? /admin-1/pkcs12/PK-1024
Writing pkcs12 file to ftp://10.1.1.1//admin-1/pkcs12/PK-1024

Writing /admin-1/pkcs12/PK-1024 !!
CRYPTO_PKI: Exported PKCS12 file successfully.
```

After you import the PKCS12 file, see the “Verifying Certificates and Trustpoints” section on page 3-27 to verify the certificate and trustpoint information.

### Importing and Exporting PEM Files

**Note**

The `crypto ca import pem` command imports only the private key (.prv), the server certificate (.crt), and the issuer certificate authority certificate (.ca). If you have more than one level of certificate authority in the certificate chain, you need to import the root and subordinate certificate authority certificates before this command is issued for authentication. Use cut-and-paste or TFTP to import the root and subordinate certificate authority certificates.

**Note**

Imported key pairs cannot be exported.

**Note**

If you are using SSH, we recommend using SCP (secure file transfer) when importing or exporting PEM files. SCP authenticates the host and encrypts the transfer session.
To import or export PEM files, perform one of these tasks:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>crypto ca import</code></td>
<td>Imports PEM files.</td>
</tr>
<tr>
<td><code>crypto ca export</code></td>
<td>Exports PEM files.</td>
</tr>
</tbody>
</table>

Note: You do not need to configure a trustpoint before importing the PEM files. Importing keys and certificates from PEM files creates the trustpoint automatically, if it does not already exist.

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>crypto ca import</code></td>
<td>Imports PEM files.</td>
</tr>
<tr>
<td><code>crypto ca export</code></td>
<td>Exports PEM files.</td>
</tr>
</tbody>
</table>

Note: Only the key, the server certificate, and the issuer certificate authority of the server certificate are exported. All higher level certificate authorities need to be exported using cut-and-paste of TFTP.

1. You will receive an error if you enter the pass phrase incorrectly.
2. A pass phrase protects a PEM file that contains a private key. The PEM file is encrypted by DES or 3DES. The encryption key is derived from the pass phrase. A PEM file containing a certificate is not encrypted and is not protected by a pass phrase.

This example shows how to import PEM files using TFTP:

```bash
ssl-proxy(config)# crypto ca import TP5 pem url tftp://10.1.1.1/TP5 password
% Importing CA certificate...
Address or name of remote host [10.1.1.1]? 10.1.1.1
Destination filename [TP5.ca]? TP5.ca
Reading file from tftp://10.1.1.1/TP5.ca
Loading TP5.ca from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 1976 bytes]

% Importing private key PEM file...
Address or name of remote host [10.1.1.1]? 10.1.1.1
Destination filename [TP5.prv]? TP5.prv
Reading file from tftp://10.1.1.1/TP5.prv
Loading TP5.prv from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 963 bytes]

% Importing certificate PEM file...
Address or name of remote host [10.1.1.1]? 10.1.1.1
Destination filename [TP5.crt]? TP5.crt
Reading file from tftp://10.1.1.1/TP5.crt
Loading TP5.crt from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 1692 bytes]

% PEM files import succeeded.
ssl-proxy(config)# end

*Apr 11 15:11:29.901: %SYS-5-CONFIG_I: Configured from console by console
```

The TP5.ca, TP5.prv, and TP5.crt files should be present on the server.
This example shows how to export PEM files using TFTP:

```bash
ssl-proxy(config)# crypto ca export TP9 pem url tftp://10.1.1.1/tp99 3des password
% Exporting CA certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.ca]?
% File 'tp99.ca' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.ca!
% Key name: key1
Usage: General Purpose Key
% Exporting private key...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.prv]?
% File 'tp99.prv' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.prv!
% Exporting router certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.crt]?
% File 'tp99.crt' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.crt!
```

After you import the PEM files, see the “Verifying Certificates and Trustpoints” section on page 3-27 to verify the certificate and trustpoint information.

### Example of Importing PEM Files for Three Levels of Certificate Authority

In this section, the root certificate authority certificate (Tier 1) and intermediate certificate authority certificate (Tier 2) are obtained using the cut-and-paste option of offline enrollment. The intermediate certificate authority certificate (Tier 3), private keys, and router certificate are obtained by importing PEM files.

1. Use cut-and-paste to obtain the root certificate authority-tier 1 certificate:

```bash
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(config)# crypto ca trustpoint 3tier-root enrollment terminal
ssl-proxy(config)# crypto ca trustpoint 3tier-root crl optional
ssl-proxy(config)# crypto ca trustpoint 3tier-root exit
ssl-proxy(config)# crypto ca authenticate 3tier-root
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGAgIBAgIQadUxzU/i97hDeZRYJlbBcDANBgkqhkiG9w0BAQUFADB1
MQswCyDyVQQDEwJVUxZTENBEGA1UEBhMBV2FsaW5vcm5pYTERMA8GA1UEBxMIc2Ft
IGpvci2IUxJMANBgNVBAYTAREFMB0wNwYDVR0PBAQDAgIAMB0tMBsGCSqGSIb3DQ
xTDAkIBAQEYGIh6x0YV3MrQlbXk0V5QyURSmIiilS9wPjSDFzPMQJIDBAQUA4G
Ko590zgRj2A90Bc14YhGgP1jH2Q7K90218gS8ZaH64KzKc5lXt//9G8BbDHw
-----END CERTIFICATE-----
```

Enter the base 64 encoded CA certificate. End with a blank line or the word "quit" on a line by itself.
Chapter 3      Configuring the SSL Services Module

Configuring Public Key Infrastructure

EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E 2C9E891 0DC3AAAA B474C7C10

% Do you accept this certificate? [yes/no]:
yes
% Certificate successfully accepted.

2. Use cut-and-paste to obtain the subordinate certificate authority 1 certificate:

ssl-proxy(config)# crypto ca trustpoint 3tier-subca1
ssl-proxy(config)# enroll terminal
ssl-proxy(config)# exit
ssl-proxy(config)# crypto ca authenticate 3tier-subca1

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

-----BEGIN CERTIFICATE-----
MIIETzCCA/mgAwIBAgIKGj0cBwAAAAAADjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQBEjVfVzETMBSGA1UDBBMKY2PsawZvcm5fYTERMA8GA1UEBxM1c2FuQgvce2UX
DjAMBgNVBAgTDWNpBObnpwc2NVMwqWYVQQLBwoNcoc3MkIiAeBgNVBAMTF3pNPsbXb24t
ZGV2dGVzdC1yd1b29LNWNM84DXTA2NTEwMzIyMTQyMVow

Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.

Trustpoint CA certificate accepted.

% Certificate successfully imported

3. Import the subordinate certificate authority 2 certificate, the RSA key pair, and router certificate.
The router certificate should be signed by subordinate certificate authority 2.

ssl-proxy(config)# crypto ca import tp-proxy1 pem terminal cisco

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

Certificate has the following attributes:
Fingerprint:50A986F6 084986F6 84986F6 84986F6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
% Certificate successfully imported.
% Enter PEM-formatted encrypted private key.
% End with "quit" on a line by itself.
-----BEGIN RSA PRIVATE KEY-----
Proc-Type:4,ENCRYPTED
DEK-Info:DES-EDE3-CBC,F0D3269840071CF8

-----BEGIN CERTIFICATE-----

% Enter PEM-formatted certificate.
% End with a blank line or "quit" on a line by itself.
-----BEGIN CERTIFICATE-----
Chapter 3 Configuring the SSL Services Module

### Configuring Public Key Infrastructure

```plaintext
% PEM files import succeeded.

ssl-proxy(config)# *Z
ssl-proxy#
* Dec 4 18:11:49.850:%SYS-5-CONFIG_I: Configured from console by console

4. Display the certificate information (optional):

```plaintext
ssl-proxy# show crypto ca certificates tp-proxy1
Certificate
Status: Available
Certificate Serial Number: 04A0147B000000000001E
Certificate Usage: General Purpose
Issuer:
  CN = sub3ca
  C = US
Subject:
  Name: ssl-proxy.
  Serial Number: B0FFF0C2
  OID.1.2.840.113549.1.9.2 = ssl-proxy.
  OID.2.5.4.5 = B0FFF0C2
CRL Distribution Point:
  http://sample.cisco.com/sub3ca.crl
Validity Date:
  start date: 18:04:09 UTC Jan 23 2003
  end   date: 21:05:17 UTC Dec 12 2003
  renew date: 00:00:00 UTC Apr 1 2003
Associated Trustpoints: tp-proxy1

CA Certificate
Status: Available
Certificate Serial Number: 6DE6B00000000007
Certificate Usage: Signature
Issuer:
  CN = sub3ca
  C = US
Subject:
  CN = sub3ca
  C = US
CRL Distribution Point:
  http://sample.cisco.com/subtest.crl
Validity Date:
  end   date: 21:05:17 UTC Dec 12 2003
Associated Trustpoints: tp-proxy1

ssl-proxy# show crypto ca certificates 3tier-subca1
CA Certificate
Status: Available
Certificate Serial Number: 29A47DEF00000000004E9
Certificate Usage: Signature
Issuer:
  CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
  C = US
Subject:
  CN = subtest
  C = US
```

---

Chapter 3 Configuring the SSL Services Module

OL-4756-01

3-26

Catalyst 6500 Series Switch SSL Services Module Configuration Note
Verifying Certificates and Trustpoints

To verify information about your certificates and trustpoints, perform this task in EXEC mode:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Displays information about the certificates associated with the specified trustpoint, or all of your certificates, the certificates of the certificate authority, and registration authority certificates.</td>
</tr>
<tr>
<td>Step 2</td>
<td>Displays information about all trustpoints or the specified trustpoint.</td>
</tr>
</tbody>
</table>

Sharing Keys and Certificates

The SSL Services Module supports the sharing of the same key pair by multiple certificates. However, this is not a good practice because if one key pair is compromised, all the certificates must be revoked and replaced.

Because proxy services are added and removed at different times, the certificates also expire at different times. Some certificate authorities require you to refresh the key pair at the time of renewal. If certificates share one key pair, you need to renew the certificates at the same time. In general, it is easier to manage certificates if each certificate has its own key pair.

The SSL Module does not impose any restrictions on sharing certificates among multiple proxy services and multiple SSL Services Modules. The same trustpoint can be assigned to multiple proxy services.

From a business point of view, the certificate authority may impose restrictions (for example, on the number of servers in a server farm that can use the same certificate). There may be contractual or licensing agreements regarding certificate sharing. Consult with the certificate authority or the legal staff regarding business contractual aspects.
In practice, some web browsers compare the subject name of the server certificate with the hostname or the IP address that appears on the URL. In case the subject name does not match the hostname or IP address, a dialog box appears, prompting the user to verify and accept the certificate. To avoid this step, limit the sharing of certificates based on the hostname or IP address.

### Saving Your Configuration

⚠️ **Caution**

RSA key pairs are saved only to NVRAM. RSA keys are *not* saved with your configuration when you specify any other file system with the `copy system:running-config file_system:` command.

Always remember to save your work when you make configuration changes.

To save your configuration to NVRAM, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy# copy</td>
<td>Saves the configuration, key pairs, and certificate to NVRAM. The key pairs are stored in the private configuration file, and each certificate is stored as a binary file in NVRAM. On bootup, the module will not need to query the certificate authority to obtain the certificates or to auto-enroll.</td>
</tr>
<tr>
<td>[ /erase ] system:running-config</td>
<td>For security reasons, we recommend that you enter the /erase option to erase the public and the private configuration files before updating the NVRAM. If you do not enter the /erase option, the key pairs from the old private configuration file may remain in the NVRAM.</td>
</tr>
<tr>
<td>nvram:startup-config</td>
<td>Note: When you enter the /erase option, both the current and the backup buffers in NVRAM are erased before the running configuration is saved into NVRAM. If a power failure or reboot occurs after the buffers are erased, but before the running configuration is saved, both configurations might be lost.</td>
</tr>
</tbody>
</table>

⚠️ **Caution**

If you have a large number of files in NVRAM, this task may take up to 2 minutes to finish.

The automatic backup of the configuration to NVRAM feature automatically backs up the last saved configuration. If the current write process fails, the configuration is restored to the previous configuration automatically.
Oversized Configuration

If you save an oversized configuration with more than 256 proxy services and 356 certificates, you might encounter a situation where you could corrupt the contents in the NVRAM.

We recommend that you always copy to running-config before saving to NVRAM. When you save the running-config file to a remote server, each certificate is saved as a hex dump in the file. If you copy the running-config file back to running-config and then save it to NVRAM, the certificates are saved again, but as binary files. However, if you copy the running-config file directly from the remote server to startup-config, the certificates saved as hex dumps are also saved, resulting in two copies of the same certificate: one in hex dump and one as a binary file. This action is unnecessary, and if the remote file is very large, it may overwrite part of the contents in the NVRAM, which could corrupt the contents.

Verifying the Saved Configuration

To verify the saved configuration, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy# show startup-config</td>
<td>Displays the startup configuration.</td>
</tr>
<tr>
<td>ssl-proxy# directory nvram:</td>
<td>Displays the names and sizes of the files in NVRAM.</td>
</tr>
</tbody>
</table>

Note: With the maximum number or proxy services (256) and certificates (356) configured, the output takes up to seven minutes to display.

Erasing the Saved Configuration

To erase a saved configuration, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy# erase nvram:</td>
<td>Erases the startup configuration and the key pairs.</td>
</tr>
<tr>
<td>ssl-proxy# erase /all nvram:</td>
<td>Erases the startup configuration, the key pairs, the certificates, and all other files from the NVRAM.</td>
</tr>
</tbody>
</table>

Note: If you have a large number of files in NVRAM, this task may take up to 2 minutes to finish.

Caution: If you erase the saved configuration, the automatic backup configuration in NVRAM is also erased.
Backing Up Keys and Certificates

If an event occurs that interrupts the process of saving the keys and certificates to NVRAM (for example, a power failure), you could lose the keys and certificates that are being saved. You can obtain public keys and certificates from the certificate authority. However, you cannot recover private keys.

If a secure server is available, back up key pairs and the associated certificate chain by exporting each trustpoint to a PKCS12 file. You can then import the PKCS12 files to recover the keys and certificates.

Security Guidelines

When backing up keys and certificates, observe the following guidelines:

- For each PKCS12, you must select a pass phrase that cannot be easily guessed and keep the pass phrase well protected. Do not store the PKCS12 file in clear form.
- The backup server must be secure. Allow only authorized personnel to access the backup server.
- When importing or exporting the PKCS12 file (in which you are required to enter a pass phrase), connect directly to the module console or use an SSH session.
- Use SCP for file transfer.

Monitoring and Maintaining Keys and Certificates

The following tasks in this section are optional:

- Deleting RSA Keys from the Module, page 3-30
- Viewing Keys and Certificates, page 3-31
- Deleting Certificates from the Configuration, page 3-31

Deleting RSA Keys from the Module

Caution

Deleting the SSH key will disable SSH on the module. If you delete the SSH key, generate a new key. See the “Configuring SSH” section on page 2-4.

Under certain circumstances you might want to delete the RSA keys from a module. For example, if you believe the RSA keys were compromised in some way and should no longer be used, you should delete the keys.

Note

When a private key is deleted while applied to an active SSL proxy service, the service will automatically enter a graceful rollover state in which it will continue to accept connections. The deleted key will stay in memory until a new key has been uploaded.
To delete all RSA keys from the module, perform this task in global configuration mode:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ssl-proxy(config)# crypto key zeroize rsa [key-label]</code></td>
<td>Deletes all RSA key pairs, or the specified key pair.</td>
</tr>
</tbody>
</table>

**Caution** If a key is deleted, all certificates that are associated with the key are deleted.

After you delete the RSA keys from a module, complete these two additional tasks:

- Ask the certificate authority administrator to revoke the certificates for your module at the certificate authority; you must supply the challenge password that you created for that module with the `crypto ca enroll` command when you originally obtained the certificates.
- Manually remove the trustpoint from the configuration, as described in the “Deleting Certificates from the Configuration” section on page 3-31.

### Viewing Keys and Certificates

To view keys and certificates, enter these commands in EXEC mode:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ssl-proxy# show crypto key mypubkey rsa</code></td>
<td>Displays RSA public keys for the module.</td>
</tr>
<tr>
<td><code>ssl-proxy# show crypto ca certificates [trustpoint_label]</code></td>
<td>Displays information about the certificate, the certificate authority certificate, and any registration authority certificates.</td>
</tr>
<tr>
<td><code>ssl-proxy# show running-config [brief]</code></td>
<td>Displays the public keys and the certificate chains. If the <code>brief</code> option is specified, the hex dump of each certificate is not displayed.</td>
</tr>
<tr>
<td><code>ssl-proxy# show ssl-proxy service proxy-name</code></td>
<td>Displays the key pair and the serial number of the certificate chain used for a specified proxy service.</td>
</tr>
</tbody>
</table>

**Note** The `proxy-name` value is case-sensitive.

### Deleting Certificates from the Configuration

The module saves its own certificates and the certificate of the certificate authority (trustpoint). You can delete certificates that are saved on the module.

**Note** When a certificate or trustpoint is deleted while applied to an active SSL proxy service, the service will automatically enter a graceful rollover state in which it will continue to accept connections. The deleted certificate or trustpoint will stay in memory until a new certificate or trustpoint has been uploaded.
To delete the certificate from the module configuration, removing the trustpoint, perform this task in global configuration mode:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy(config)# no crypto ca trustpoint trustpoint-label</td>
<td>Deletes the certificate and trustpoint.</td>
</tr>
</tbody>
</table>

### Assigning a Certificate to a Proxy Service

When you enter the `certificate rsa general-purpose trustpoint trustpoint_label` subcommand (under the `ssl-proxy service proxy_service` command), you assign a certificate to the specified proxy service. You can enter the `certificate rsa general-purpose trustpoint` subcommand multiple times for the proxy service.

If the trustpoint label is modified, the proxy service is momentarily taken out of service during the transition. Existing connections continue to use the old certificate until the connections are closed or cleared. New connections use the certificate from the new trustpoint, and the service is available again.

However, if the new trustpoint does not have a certificate yet, the operational status of the service remains down. New connections are not established until the new certificate is available. If the certificate is deleted by entering the `no certificate rsa general-purpose trustpoint` subcommand, the existing connections continue to use the certificate until the connections are closed or cleared. Although the certificate is obsolete, it is not removed from the proxy service until all connections are closed or cleared.

**Note**

You can assign a generated self-signed certificate to a proxy service, but you cannot assign an imported self-signed certificate to a proxy service, because you cannot import the key pair of the certificate authority that signed the imported certificate. See the “Generating a Self-Signed Certificate” section on page C-1 for more information on self-signed certificates.

The following example shows how to assign a trustpoint to a proxy service:

```plaintext
ssl-proxy# configure terminal
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# virtual ip 10.1.1.2 p tcp p 443
ssl-proxy(config-ssl-proxy)# server ip 20.0.0.3 p tcp p 80
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# certificate rsa general trustpoint tp-1
ssl-proxy(config-ssl-proxy)# end
ssl-proxy# show ssl-proxy service s2
Service id:6, bound_service_id:262
Virtual IP:10.1.1.2, port:443
Server IP:20.0.0.3, port:80
rsa-general-purpose certificate trustpoint:tp-1
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:tp-1
    Serial Number:3C2CD2330001000000DB
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
ssl-proxy#
```
Chapter 3  Configuring the SSL Services Module

Configuring Public Key Infrastructure

The following example shows how to change a trustpoint for a proxy service:

```plaintext
Note
The existing connections continue to use the old certificate until the connections are closed. The operational status of the service changes from up to down, and then up again. New connections use the new certificate.

ssl-proxy# configure terminal
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# certificate rsa general trustpoint tp-2
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#
ssl-proxy# show ssl-proxy service s2
Service id:6, bound_service_id:262
Virtual IP:10.1.1.2, port:443
Server IP:20.0.0.3, port:80
rsa-general-purpose certificate trustpoint:tp-2
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:k2
    Serial Number:70FCBFEC0001000000D65
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Obsolete certificate chain in use for old connections:
  Server Certificate:
    Key Label:tp-1
    Serial Number:3C2CD2330001000000DB
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
ssl-proxy#  
```

Renewing a Certificate

Some certificate authorities require you to generate a new key pair to renew a certificate, while other certificate authorities allow you to use the key pair of the expiring certificate to renew a certificate. Both cases are supported on the SSL Services Module.

The SSL server certificates usually expire in one or two years. Graceful rollover of certificates avoids sudden loss of services.

In the following example, proxy service s2 is assigned trustpoint t2:

```plaintext
ssl-proxy# configure terminal
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint t2
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#
ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
Certificate chain in use for new connections:
  Server Certificate:
```

Key Label:k2
Serial Number:1DFBB1FD0001000000D48
Root CA Certificate:
  Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up

In the following example, the key pair for trustpoint t2 is refreshed, and the old certificate is deleted from the Cisco IOS database. Graceful rollover starts automatically for proxy service s2.

```bash
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# crypto key generate rsa general-key k2 exportable
% You already have RSA keys defined named k2.
% Do you really want to replace them? [yes/no]:yes
Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose Keys. Choosing a key modulus greater than 512 may take a few minutes.
```
How many bits in the modulus [512]: **1024**
% Generating 1024 bit RSA keys ...[OK]
ssl-proxy(config)# end
ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
Certificate chain in graceful rollover, being renewed:
  Server Certificate:
    Key Label:k2
    Serial Number:1DFBB1FD000100000D48
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE626895305511AC4
Server certificate in graceful rollover
Admin Status:up
Operation Status:up

In the following example, existing and new connections use the old certificate until trustpoint t2 reenrolls. After trustpoint t2 reenrolls, new connections use the new certificate; existing connections continue to use the old certificate until the connections are closed.

global# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
global(config)# crypto ca enroll t2
% Start certificate enrollment . .
% The subject name in the certificate will be:CN=host1.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:00000000
% The IP address in the certificate is 10.1.1.1
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

  Fingerprint: 6518C579 A0498063 C5795057 A6170 075

global(config)# end
*Sep 24 15:19:34.339:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority

ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:k2
    Serial Number:2475A2FC000100000D4D
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626895305511AC4
Obsolette certificate chain in use for old connections:
  Server Certificate:
    Key Label:k2
    Serial Number:1DFBB1FD000100000D48
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626895305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
In the following example, the obsolete certificate is removed after all of the existing connections are closed.

```bash
ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
    Certificate chain in use for new connections:
    Server Certificate:
        Key Label:k2
        Serial Number:2475A2FC000100000D4D
        Root CA Certificate:
            Serial Number:313AD6510D25ABE46268E96305511AC4
    Certificate chain complete
    Admin Status:up
    Operation Status:up
```

### Automatic Certificate Renewal and Enrollment

When you configure automatic enrollment, the module automatically requests a certificate from the certificate authority that is using the parameters in the configuration.

You can configure the certificate to automatically renew after a specified percentage of the validity time has passed. For example, if the certificate is valid for 300 days, and you specify `renewal_percent` as 80, the certificate automatically renews after 240 days have passed since the start validity time of the certificate.

Note
The certificate authority certificate needs to be in the database prior to auto enrollment or renewal. Authenticate the trustpoint prior to configuring automatic enrollment. Also, configure a SCEP enrollment URL for the trustpoint.

To enable automatic enrollment and renewal and to display timer information, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>ssl-proxy(config)# crypto ca trustpoint trustpoint-label</code></td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Declares the trustpoint.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>`ssl-proxy(ca-trustpoint)# auto-enroll {renewal_percent</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Enables automatic renewal and enrollment for the specified trustpoint.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>Valid values for <code>renewal_percent</code> are 0 (enroll within 1 minute) through 100.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>The <code>regenerate</code> keyword generates a new key for the certificate even if a named key already exists.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><code>ssl-proxy# show crypto ca timers</code></td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Displays the time remaining before each timer expires.</td>
</tr>
</tbody>
</table>
This example shows how to enable auto enrollment and auto renewal:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# crypto ca trustpoint tk21
ssl-proxy(ca-trustpoint)# auto-enroll 90
ssl-proxy(ca-trustpoint)# end
ssl-proxy# show crypto ca timers
PKI Timers
          44.306
          44.306  RENEW tp-new
255d 5:28:32.348  RENEW tk21
ssl-proxy#
```

### Enabling Key and Certificate History

When you enter the `ssl-proxy pki history` command, you enable the SSL proxy services key and certificate history. This history creates a record for each addition or deletion of the key pair and certificate chain for a proxy service.

When you enter the show `ssl-proxy certificate-history` command, the records are displayed. Each record logs the service name, key pair name, time of generation or import, trustpoint name, certificate subject name and issuer name, serial number, and date.

You can store up to 512 records in memory. For each record, a syslog message is generated. The oldest records are deleted after the limit of 512 records is reached.

To enable key and certificate history and display the records, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>ssl-proxy(config)# ssl-proxy pki history</code> Enables key and certificate history.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>ssl-proxy# show ssl-proxy certificate-history [service proxy_service]</code> Displays key and certificate history records for all services or the specified service.</td>
</tr>
</tbody>
</table>

This example shows how to enable the key and certificate history and display the records for a specified proxy service:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy pki history
ssl-proxy(config)# end
ssl-proxy# show ssl-proxy certificate-history service s2
Record 1, Timestamp:00:00:22, 17:44:18 UTC Sep 29 2002
  Installed Server Certificate, Index 0
  Proxy Service:s2, Trust Point:t2
  Key Pair Name:k2, Key Usage:RSA General Purpose, Not Exportable
  Time of Key Generation:06:29:08 UTC Sep 28 2002
  Subject Name:CN = host1.cisco.com, OID.1.2.840.113549.1.9.2 = ssl-proxy.cisco.com,
  OID.1.2.840.113549.1.9.8 = 10.1.1.1
  Issuer Name:CN = TestCA, OU = Lab, O = Cisco Systems, L = San Jose, ST = CA, C = US,
  EA =<16> simpson-pki@cisco.com
  Serial Number:3728ADCD000100000D4F
  Validity Start Time:15:56:55 UTC Sep 28 2002
  End Time:16:06:55 UTC Sep 28 2003
  Renew Time:00:00:00 UTC Jan 1 1970
End of Certificate Record
Total number of certificate history records displayed = 1
```
**Caching Peer Certificates**

You can configure the SSL module to cache authenticated server and client (peer) certificates. Caching authenticated peer certificates saves time by not requiring the SSL module to authenticate the same certificate again. The SSL module uses the cached certificate information when it receives the same peer certificate within a specified timeout interval and the verify option (signature-only or all) matches.

---

**Note**

When specifying **verifying all**, the CRL lookup or an ACL filtering result could change within the specified timeout interval, causing the SSL module to incorrectly accept a certificate that should be denied. For instance, the SSL module could cache a peer certificate, then, within the specified timeout, download an updated CRL in which the certificate is listed.

---

In an environment where the SSL module repeatedly receives a number of certificates and the risk is acceptable, caching authenticated peer certificates can reduce the overhead.

For example, in a site-to-site VPN environment, where two SSL modules authenticate each other’s certificates during full handshakes, caching is applicable. A combination of verifying signature-only and caching authenticated peer certificates gives the best performance.

Peer certificates that expire within the specified time interval are not cached. The SSL module uses separate cache entries for signature-only and verify-all options. Matching the verify options is one of the criteria for a cache hit.

Since the same peer certificate can be received on different proxy services at different times, and each proxy service has its own certificate authority pool, the issuer of the peer certificate may be in the certificate authority pool of the previous proxy service and not in the certificate authority pool of the current proxy service. This is considered a cache miss, and the peer certificate is verified.

To cache authenticated peer certificates, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ssl-proxy(config)# ssl-proxy pki cache size size timeout minutes</code></td>
<td>Configures the certificate cache parameters. The default value for <code>size</code> is 0 (disabled); valid values are 0 through 5000 entries. The default value for <code>minutes</code> is 15 minutes; valid values are 1 through 600 minutes. To clear the cache, change the cache size or the timeout value.</td>
</tr>
</tbody>
</table>

**Configuring Certificate Expiration Warning**

You can configure the SSL module to log warning messages when certificates have expired or will expire within a specified amount of time. When you enable certificate expiration warnings, the SSL module checks every 30 minutes for the expiration information for the following:

- all the proxy services
- the certificate authority certificates associated with the proxy services
- all of the certificate authority trustpoints that are assigned to the trusted certificate authority pools

You can specify a time interval between 1 and 720 hours.
The SSL module stores information about which certificates have been logged. Specifying 0 disables the warnings and clears the internal memory of any previously logged warning message. Specifying a time interval between 1 and 720 hours restarts the logging process. Log messages may not be displayed immediately after you restart logging. You may have to wait up to 30 minutes to see the first log message.

If a certificate will expire within the specified interval, or has already expired, the SSL module logs a single warning message for the certificate.

In addition, you can enable the CISCO-SSL-PROXY-MIB certificate expiration trap to issue a trap each time a proxy service certificate expiration warning is logged. For more information on configuring SNMP traps, see the “Configuring SNMP Traps” section on page 4-14.

To enable the certificate expiration warning and configure the warning interval, perform this task:

To enable the certificate expiration warning and configure the warning interval, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval</td>
<td>Enables certificate expiration warning and configures the warning interval. The default value for interval is 0 (disabled); valid values are 1 though 720 hours.</td>
</tr>
<tr>
<td>Note: Entering 0 disables warnings and clears the internal memory of any previously logged warning messages. No SNMP traps are sent.</td>
<td></td>
</tr>
</tbody>
</table>

This example shows how to enable the certificate expiration warning and configure the warning interval:

ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 36
*Nov 27 03:44:05.207:%STE-6-PKI_CERT_EXP_WARN_ENABLED:Proxy service certificate expiration warning has been enabled. Time interval is set to 36 hours.
ssl-proxy(config)# end

This example shows how to clear the internal memory of any previously logged warning messages and restart the logging process:

ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 0
*Nov 27 03:44:15.207:%STE-6-PKI_CERT_EXP_WARN_DISABLED:Checking of certificate expiration has been disabled.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 1
*Nov 27 03:44:16.207:%STE-6-PKI_CERT_EXP_WARN_ENABLED:Proxy service certificate expiration warning has been enabled. Time interval is set to 36 hours.
ssl-proxy(config)# end

This example shows a syslog message for a proxy service certificate that will expire or has expired:

Jan 1 00:00:18.971:%STE-4-PKI_PROXY_SERVICE_CERT_EXPIRING:A proxy service certificate is going to expire or has expired at this time:20:16:26 UTC Sep 5 2004, proxy service:s7, trustpoint:tk21.
This example shows a syslog message for a certificate authority certificate that is associated with one or more proxy services that will expire or has expired:

Jan 1 00:00:18.971:%STE-4-PKI_PROXY_SERVICE_CA_CERT_EXPIRING:A CA certificate is going to expire or has expired at this time:22:19:38 UTC Mar 4 2004, subject name:CN = ExampleCA, OU = Example Lab, O = Cisco Systems, L = San Jose, ST = CA, C = US, EA = example@cisco.com, serial number:313AD6510D25ABAE4626E96305511AC4.

This example shows a syslog message for a certificate authority certificate assigned to a trusted certificate authority pool that will expire or has expired:

Jan 1 00:00:18.971:%STE-4-PKI_CA_POOL_CERT_EXPIRING:A CA certificate in a CA pool is going to expire or has expired at this time:22:19:38 UTC Mar 4 2004, CA pool:pool2, trustpoint:tp1-root.

### Configuring SSL Proxy Services

You define SSL proxy services using the `ssl-proxy service ssl_proxy_name` command. You can configure the virtual IP address and port associated with the proxy service, and the associated target IP address and port. You define TCP and SSL policies for both client (`virtual`) and server (`server`) sides of the proxy.

The following sections describe how to configure proxy services:
- SSL Server Proxy Services, page 3-40
- SSL Client Proxy Services, page 3-43

#### SSL Server Proxy Services

To configure SSL server proxy services, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**  
ssl-proxy(config)# ssl-proxy service service_name | Defines the name of the SSL proxy service.  
*Note* You cannot use the same `service_name` for both the server proxy service and the client proxy service.  
*Note* The `service_name` value is case-sensitive. |
| **Step 2**  
ssl-proxy(config-ssl-proxy)# virtual ipaddr ip_addr [mask_addr] {protocol tcp port port} [secondary^1,^2,^4] | Defines the virtual server IP address, transport protocol (TCP), and port number for which the SSL Services Module is the proxy.  
*Note* The `secondary` keyword is required if the virtual IP address is not on a network with a direct connection. |
| **Step 3**  
ssl-proxy(config-ssl-proxy)# server ipaddr ip_addr protocol tcp port port | Defines the IP address, port number, and the transport protocol of the target server for the proxy.  
*Note* The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server. |
| **Step 4**  
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp_policy_name^5 | (Optional) Applies a TCP policy to the client side of the proxy server. See the “Configuring TCP Policy” section on page 4-4 for TCP policy parameters. |
### Command Purpose

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl_policy_name</td>
<td>(Optional) Applies an SSL policy to the client side of the proxy server. See the “Configuring SSL Policy” section on page 4-2 for SSL policy parameters.</td>
</tr>
<tr>
<td>6</td>
<td>ssl-proxy(config-ssl-proxy)# server policy tcp tcp_policy_name</td>
<td>(Optional) Applies a TCP policy to the server side of the proxy server. See the “Configuring TCP Policy” section on page 4-4.</td>
</tr>
<tr>
<td>7</td>
<td>ssl-proxy(config-ssl-proxy)# policy http-header http_header_policy_name</td>
<td>(Optional) Applies the HTTP header policy to proxy server. See the “HTTP Header Insertion” section on page 4-6.</td>
</tr>
<tr>
<td>8</td>
<td>ssl-proxy(config-ssl-proxy)# policy url-rewrite url_rewrite_policy_name</td>
<td>(Optional) Applies the URL rewrite policy. See the “Configuring URL Rewrite” section on page 4-10.</td>
</tr>
<tr>
<td>9</td>
<td>ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name</td>
<td>(Optional) Associates the trusted certificate authority pool with the proxy service. See the “Server Certificate Authentication” section on page 3-49 for information on certificate authority pools.</td>
</tr>
<tr>
<td>10</td>
<td>ssl-proxy(config-ssl-proxy)# authenticate verify (signature-only</td>
<td>(Optional) Enables server certificate authentication and specifies the form of verification. See the “Server Certificate Authentication” section on page 3-49 for information on server certificate authentication.</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>)</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>)</td>
</tr>
<tr>
<td>11</td>
<td>ssl-proxy(config-ssl-proxy)# nat (server</td>
<td>(Optional) Specifies the usage of either server NAT (network address translation) or client NAT for the server-side connection opened by the SSL Services Module. See the “Configuring NAT” section on page 4-12.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>client natpool_name)</td>
</tr>
<tr>
<td>12</td>
<td>ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint trustpoint_label</td>
<td>Applies a trustpoint configuration to the proxy server. The trustpoint defines the certificate authority server, the key parameters and key-generation methods, and the certificate enrollment methods for the proxy server. See the “Declaring the Trustpoint” section on page 3-7 for information on configuring the trust point.</td>
</tr>
<tr>
<td>13</td>
<td>ssl-proxy(config-ssl-proxy)# inservice</td>
<td>Sets the proxy server as administratively Up.</td>
</tr>
</tbody>
</table>

**Note**

- The trustpoint defines the certificate authority server, the key parameters and key-generation methods, and the certificate enrollment methods for the proxy server. See Chapter 5, “Configuring Different Modes of Operation,” for information on configuring the SSL Services Module with the CSM.

- You can enter the `secondary` keyword when the SSL Services Module is used in a standalone configuration or when the SSL Services Module is used as a real server on a load balancer (like the CSM) configured in dispatch mode (MAC address rewrite). See Chapter 5, “Configuring Different Modes of Operation,” for information on configuring the SSL Services Module with the CSM.

- If you create a policy without specifying any parameters, the policy is created using the default values.

1. Configure the mask address to specify a wildcard proxy service. You must enter the `secondary` keyword to configure a wildcard proxy service.
2. When you enter the `secondary` keyword, the SSL Services Module does not respond to ARP requests of the virtual IP address.
3. You can enter the `secondary` keyword when the SSL Services Module is used in a standalone configuration or when the SSL Services Module is used as a real server on a load balancer (like the CSM) configured in dispatch mode (MAC address rewrite). See Chapter 5, “Configuring Different Modes of Operation,” for information on configuring the SSL Services Module with the CSM.
4. You can enter the `secondary` keyword if you configure multiple devices using the same virtual IP address. The virtual IP address can be any legal IP address, and does not have to be in the VLAN (subnet) connected to the SSL Services Module.
5. If you create a policy without specifying any parameters, the policy is created using the default values.
6. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
Chapter 3  Configuring the SSL Services Module

Configuring SSL Proxy Services

This example shows how to configure SSL proxy services:

```plaintext
ssl-proxy(config)# ssl-proxy service proxy1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ssl-proxy)# nat client t2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint tp1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

If you have many virtual and server IP addresses to manage and configure, you can configure a wildcard proxy service.

This example shows how to configure a wildcard SSL proxy service, so that `proxy1` accepts virtual IP addresses 10.0.0.1 through 10.255.255.254:

```plaintext
ssl-proxy(config)# ssl-proxy service proxy1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.0.0.0 255.0.0.0 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 20.1.2.3 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ssl-proxy)# nat client t2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint tp1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

**SSL Version 2.0 Forwarding**

The SSL module is not able to terminate SSL version 2.0 (SSLv2) connections. However, you can configure the SSL module to forward SSLv2 connections to another server by entering the `sslv2` keyword at the `server` command. When you configure the SSLv2 server IP address, the SSL module transparently forwards all SSLv2 connections to that server. If you require SSLv2 forwarding, you need to configure the SSLv2 server IP address in addition to the IP address of the server that is used for offloading SSL version 3.0 or Transport Layer Security (TLS) version 1.0 connections.

To configure SSLv2 forwarding, enter this command:

```plaintext
ssl-proxy(config-ssl-proxy)# server ipaddr ip_addr protocol tcp port port sslv2
```

Defines the IP address, port number, and the transport protocol of the target server for the proxy.

**Note**  The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server.

1. Enter the `sslv2` keyword to forward SSL version 2.0 client connections to a SSL v2.0 server. When you enter `sslv2`, configure another server IP address to offload SSL version 3.0 or Transport Layer Security (TLS) version 1.0 connections.
This example shows how to configure SSL proxy services to forward SSL v2.0 connections:

```plaintext
ssl-proxy(config)# ssl-proxy service frontend
ssl-proxy(config-ssl-proxy)# virtual ipaddr 35.200.200.102 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 26.51.51.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# server ipaddr 26.51.51.2 protocol tcp port 443 sslv2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

### SSL Client Proxy Services

You configure SSL client proxy services to specify that the proxy service accepts clear text traffic, encrypts the traffic into SSL traffic, and forwards the traffic to the backend SSL server.

While you are required to configure a certificate for the SSL server proxy, you are not required to configure a certificate for SSL client proxy. If you configure the certificate for the SSL client proxy, that certificate is sent in response to the CertificateRequest message that is sent by the server during the client authentication phase of the handshake protocol.

---

**Note**

SSL policies are configured at the `server` subcommand for SSL server proxy services; SSL policies are configured at the `virtual` subcommand for SSL server proxy services.

---

To configure SSL client proxy services, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Defines the name of the SSL proxy service. The <code>client</code> keyword configures the SSL client proxy service.</td>
</tr>
<tr>
<td><code>ssl-proxy(config)# ssl-proxy service service_name client</code></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>You cannot use the same <code>service_name</code> for both the server proxy service and the client proxy service.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>The <code>service_name</code> value is case-sensitive.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Defines the virtual server IP address, transport protocol (TCP), and port number for which the SSL Services Module is the proxy.</td>
</tr>
<tr>
<td><code>ssl-proxy(config-ssl-proxy)# virtual ipaddr ip_addr [mask_addr]1 protocol tcp port port [secondary2,3,4]</code></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>The <code>secondary</code> keyword is required if the virtual IP address is not on a network with a direct connection.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Defines the IP address, port number, and the transport protocol of the target server for the proxy.</td>
</tr>
<tr>
<td><code>ssl-proxy(config-ssl-proxy)# server ipaddr ip_addr protocol tcp port port</code></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>(Optional) Applies a TCP policy to the client side of the proxy server. See the “Configuring TCP Policy” section on page 4-4 for TCP policy parameters.</td>
</tr>
<tr>
<td><code>ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp_policy_name5</code></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>(Optional) Applies an SSL policy to the server side of the proxy server. See the “Configuring SSL Policy” section on page 4-2 for SSL policy parameters.</td>
</tr>
<tr>
<td><code>ssl-proxy(config-ssl-proxy)# server policy ssl ssl_policy_name6</code></td>
<td><strong>Purpose</strong></td>
</tr>
</tbody>
</table>
### Command

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td><code>ssl-proxy(config-ssl-proxy)# server policy tcp tcp_policy_name</code></td>
<td>(Optional) Applies a TCP policy to the server side of the proxy server. See the “Configuring TCP Policy” section on page 4-4.</td>
</tr>
<tr>
<td>7</td>
<td><code>ssl-proxy(config-ssl-proxy)# policy http-header http_header_policy_name</code></td>
<td>(Optional) Applies the HTTP header policy to proxy server. See the “HTTP Header Insertion” section on page 4-6.</td>
</tr>
<tr>
<td>8</td>
<td><code>ssl-proxy(config-ssl-proxy)# policy url-rewrite url_rewrite_policy_name</code></td>
<td>(Optional) Applies the URL rewrite policy. See the “Configuring URL Rewrite” section on page 4-10.</td>
</tr>
<tr>
<td>9</td>
<td><code>ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name</code></td>
<td>Associates the trusted certificate authority pool with the proxy service. See the “Client Certificate Authentication” section on page 3-46 for information on certificate authority pools.</td>
</tr>
<tr>
<td>10</td>
<td>`ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only</td>
<td>all}`</td>
</tr>
<tr>
<td>11</td>
<td>`ssl-proxy(config-ssl-proxy)# nat {server</td>
<td>client natpool_name}`</td>
</tr>
<tr>
<td>12</td>
<td><code>ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint trustpoint_label</code></td>
<td>(Optional) Applies a trustpoint configuration to the client proxy. <strong>Note</strong> The trustpoint defines the certificate authority server, the key parameters and key-generation methods, and the certificate enrollment methods for the proxy server. See the “Declaring the Trustpoint” section on page 3-7 for information on configuring the trust point.</td>
</tr>
<tr>
<td>13</td>
<td><code>ssl-proxy(config-ssl-proxy)# inservice</code></td>
<td>Sets the proxy server as administratively Up. <strong>Note</strong> An SSL proxy service can be configured prior to configuring the associated trustpoint or downloading the corresponding certificate, but the service will not enter the administratively Up state until a certificate is obtained.</td>
</tr>
</tbody>
</table>

1. Configure the mask address to specify a wildcard proxy service. You must enter the `secondary` keyword to configure a wildcard proxy service.
2. When you enter the `secondary` keyword, the SSL Services Module does not respond to ARP requests of the virtual IP address.
3. You can enter the `secondary` keyword when the SSL Services Module is used in a standalone configuration or when the SSL Services Module is used as a real server on a load balancer (like the CSM) configured in dispatch mode (MAC address rewrite). See Chapter 5, “Configuring Different Modes of Operation” for information on configuring the SSL Services Module with the CSM.
4. You can enter the `secondary` keyword if you configure multiple devices using the same virtual IP address. The virtual IP address can be any legal IP address, and does not have to be in the VLAN (subnet) connected to the SSL Services Module.
5. If you create a policy without specifying any parameters, the policy is created using the default values.
6. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
7. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL module authenticates all the certificates in the peer certificate chain and stops only at the highest level certificate authority. There must be a certificate authority trustpoint for the highest level certificate authority, and this trustpoint should be authenticated.
This example shows how to configure SSL client proxy services:

```bash
ssl-proxy(config)# ssl-proxy service proxy1 client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy ssl ssl1
ssl-proxy(config-ssl-proxy)# in-service
ssl-proxy(config-ssl-proxy)# end
```

**Configuring Certificate Authentication**

This section describes how to configure client and server authentication:

- **Client Certificate Authentication**, page 3-46
- **Server Certificate Authentication**, page 3-49
- **Certificate Revocation List**, page 3-52
- **Certificate Security Attribute-Based Access Control**, page 3-57

When you configure client or server certificate authentication, you need to specify the form of verification as `signature-only` or `all`. Both options check the validity start time and validity end time of each certificate being authenticated. If the start time is in the future or the end time has passed, the SSL module does not accept the certificate.

When you enter the `verify signature-only` command, the SSL module verifies the certificate chain from the peer certificate to the next certificate (which should be the issuer of the previous certificate), then to the next certificate, and so on until one of the following conditions is met:

- The certificate is issued by a trusted certificate authority, or the certificate itself matches a trusted certificate authority certificate, and the trusted certificate authority is in the certificate authority pool assigned to the proxy service. In this case, the chain is accepted, and the rest of the chain does not need to be verified.
- The end of the chain is reached, and the last certificate in the chain is not issued by a trusted certificate authority. In this case, the chain is rejected.

When you enter the `verify all` command, the SSL module sorts the certificate chain in order, ignoring any unrelated or redundant certificates. The SSL module determines if the top-most certificate in the sorted chain is issued by a trusted certificate authority or if it matches a trusted certificate authority certificate.

If the SSL module cannot trust the top-most certificate, the chain is rejected.

If the SSL module trusts the top-most certificate, then the SSL module performs the following for each certificate in the chain:

- Verifies the signature of each certificate.
- If the certificate is associated with one or more trustpoints, the SSL module selects one of these trustpoints. Depending on the CRL and ACL map configuration for this trustpoint, the SSL module performs revocation and certificate attribute filtering. If the CRL or ACL checking denies the certificate, the SSL module rejects the chain.
- If the certificate is a X509 version 3 certificate authority certificate, the SSL module verifies that the Basic Constraints extension is present and valid. If the Basic Constraints extension is not present or valid, the chain is rejected.
If you verify only the signature, that verification process checks only the validity and signature of a minimum number of certificates in the chain. Verifying all performs more checking and validates all the certificates received, but takes longer and uses more CPU time.

You can download and update CRLs by entering CLI commands to reduce real-time delay. However, CRL lookup is a slow process. See the “Certificate Revocation List” section on page 3-52 for information about CRLs.

### Client Certificate Authentication

When you configure the SSL module as an SSL server, you can configure the SSL module to authenticate the SSL client. In this case, the SSL module requests a certificate from the SSL client for authentication.

To authenticate the SSL client, the SSL module verifies the following:

- The certificate at one level is properly signed by the issuer at the next level.
- At least one of the issuer certificates in the certificate chain is trusted by the SSL proxy service.
- None of the certificates in the certificate chain is in the certificate revocation list (CRL) and rejected by any access control list (ACL).

For verifying the SSL client certificates, the SSL module is configured with a list of trusted certificate authorities (certificate authority pool). A trusted certificate authority pool is a subset of the trusted certificate authorities in the database. The SSL module trusts only the certificates issued by the certificate authorities that you configure in the certificate authority pool.

When you configure the SSL module as an SSL server, and it needs to authenticate the client’s certificate,

**Note**

For a proxy service to be operationally up, the certificate authority pool must have at least one trustpoint that has a certificate. If none of the trustpoints in the certificate authority pool has a certificate, the proxy service will go down automatically.

**Note**

Authentication may fail if a particular level of certificate authority in the hierarchy is not included in the certificate authority pool. To avoid this type of failure and to improve efficiency when verifying signature-only for authentication, add all levels of subordinate certificate authorities together with the root certificate authority into a certificate authority pool.

**Note**

If a certificate authority trustpoint is deleted, you should remove the corresponding trustpoint from the trusted certificate authority pool.

To configure client authentication, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong>&lt;br&gt;ssl-proxy(config)# ssl-proxy pool ca_pool_name</td>
<td>Creates the certificate authority pool.&lt;br&gt;Note: You can create up to eight pools.</td>
</tr>
<tr>
<td><strong>Step 2</strong>&lt;br&gt;ssl-proxy(config-ssl-proxy)# ca trustpoint ca_trustpoint_label</td>
<td>Adds a trusted certificate authority to the pool.&lt;br&gt;Note: You can add up to 16 trusted certificate authorities per pool.</td>
</tr>
</tbody>
</table>
### Chapter 3 Configuring the SSL Services Module

#### Configuring Certificate Authentication

<table>
<thead>
<tr>
<th>Step 3</th>
<th><code>ssl-proxy(config-ssl-proxy)# exit</code></th>
<th>Returns to config mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 4</td>
<td><code>ssl-proxy(config)# ssl-proxy service proxy_name</code></td>
<td>Defines the name of the SSL server proxy service.</td>
</tr>
<tr>
<td>Step 5</td>
<td><code>ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name</code></td>
<td>Associates the trusted certificate authority pool with the proxy service.</td>
</tr>
<tr>
<td>Step 6</td>
<td>`ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only</td>
<td>all}`</td>
</tr>
<tr>
<td>Step 7</td>
<td><code>ssl-proxy(config-ssl-proxy)# exit</code></td>
<td>Exits from configuration mode.</td>
</tr>
</tbody>
</table>

1. The SSL module supports up to eight levels of certificate authority. We recommend that you add all levels of certificate authority, or at least the root certificate authority, to the certificate authority pool.  
2. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.  
3. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL module authenticates all the certificates in the peer certificate chain and stops only at the highest level certificate authority. There must be a certificate authority trustpoint at the highest level certificate authority, and this trustpoint should be authenticated.

This example shows how to configure client certificate authentication verifying signature only:

```plaintext
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(config-ca-trustpoint)# enrollment terminal
ssl-proxy(config-ssl-proxy)# exit

ssl-proxy(config)# ssl-proxy service proxy_name

ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name

ssl-proxy(config-ssl-proxy)# authenticate verify 
  {signature-only | all}

ssl-proxy(config-ssl-proxy)# exit
```

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

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFAADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2Vub24lMjQNCgV3MSAyMDYwMDA1MDAwMB4GCSqGSIb3DQEBCwUAUwIBABoD
V29ybGQgQ29yc2luZwVzZW50eSBodHRwczovL2RldnRlc3Qtcm9vdC1DQS5jcmVhdG1
VcmVzc2lvbi1uYW1lLUNBLmNybDAgBgNVH REJFQ0EBAQIwgYwBAs5

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

Certificate has the following attributes:
- Fingerprint: AC6FC55E CC29E891 0DC3FAA B4747C10
- % Do you accept this certificate? [yes/no]: yes
- Trustpoint CA certificate accepted.
- % Certificate successfully imported
**Chapter 3  Configuring the SSL Services Module**

**Configuring Certificate Authentication**

**This example shows how to configure client certificate authentication verifying all:**

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAbGwIBAgIeRJ1hR7cOD9Ig5X3uOZTzoaQ85Ou7I/vHk3ZPSkYKo5bTn0
GzSxug0iXZ89zqOgDJu9Z7EzPQG71l1i850tsk+gXe0umu7SOn3L3y2Iyf7goj/Ax
mBtiDd+xglnMqsp9Q5zZv21tVz84ujQmXN7OxKZpGu8a4CuGP0JzAQy5d/1BfQ6
2983W832YQG+3v9C991D0KfN2mJjG9A1m6M10nFb1vZyajwDQh6rH0H7Oz0e/3
Sklj4skj4skj4skj4skj4skj4skj4skj4skj4skj4skj4skj4skj4skj4skj4skj4skj
-----END CERTIFICATE-----
Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

```c
ssl-proxy(config)#
ssl-proxy(config)#
ssl-proxy(config-ca-pool)# enrollment terminal

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```
Server Certificate Authentication

When you configure the SSL module as an SSL client (for example, for backend encryption), the SSL module always authenticates the SSL server.

To authenticate the SSL server, the SSL module verifies the following:

- The certificate at one level is properly signed by the issuer at the next level.
- At least one of the issuer certificates in the certificate chain is trusted by the SSL proxy service.
- None of the certificates in the certificate chain is in the certificate revocation list (CRL) and rejected by any access control list (ACL).

By default, the SSL module accepts any certificate issued by any certificate authority trustpoint that has a certificate, is not listed in the CRL, and is not rejected by an ACL.

Optionally, you can create a trusted certificate authority pool and associate it with the proxy service. In this case, the SSL module accepts only certificates issued by the certificate authorities in the pool.

You can also select to verify only the signature by entering the `authenticate verify signature-only` command. Verifying the signature skips CRL and ACL checking. You must configure a trusted certificate authorities pool in order to specify the signature-only option.

To configure server certificate authentication, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>ssl-proxy(config)# ssl-proxy pool ca_pool_name</code>&lt;br&gt;Copies the certificate authority pool. <strong>Note</strong> You can create up to eight pools.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>ssl-proxy(config-ssl-proxy)# ca trustpoint ca_trustpoint_label1</code>&lt;br&gt;Adds a trusted certificate authority to the pool. <strong>Note</strong> You can add up to 16 trusted certificate authorities per pool.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><code>ssl-proxy(config-ssl-proxy)# exit</code>&lt;br&gt;Returns to config mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><code>ssl-proxy(config)# ssl-proxy service proxy_name client</code>&lt;br&gt;Defines the name of the SSL client proxy service. <strong>Note</strong> Enter the client keyword to configure SSL client proxy services.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><code>ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name</code>&lt;br&gt;(Optional) Associates the certificate authority pool with the proxy service. <strong>Note</strong> If you specify signature-only in <strong>Step 6</strong>, you must configure a certificate authority pool.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><code>ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only1 all1}</code>&lt;br&gt;(Optional) Enables server certificate authentication and specifies the form of verification. <strong>Note</strong> The signature-only keyword verifies the signature only, without looking up the CRL or matching ACL. You must configure a certificate authority pool to specify signature-only. The default is all.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><code>ssl-proxy(config-ssl-proxy)# exit</code>&lt;br&gt;Exits configuration mode</td>
</tr>
</tbody>
</table>

1. The SSL module supports up to eight levels of certificate authority. We recommend that you add all levels of certificate authority, or at least the root certificate authority, to the certificate authority pool.
2. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
This example shows how to configure server certificate authentication verifying signature only:

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(config-ca-trustpoint)# enrollment terminal
ssl-proxy(config-ca-trustpoint)#
ssl-proxy(config-ca-trustpoint)# crypto ca authenticate rootca
```

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

```
-----BEGIN CERTIFICATE-----
MIIC1zCCCh Geometry endpoint for Cisco Catalyst 6500 Series Switch SSL Services Module Configuration Note
GFgIBy9KasB/D0MhZ4sE0IuX+5 528244ZL6dV1y/Tp92A6VQgi5Wp3z1p
bXe：<21:ZG52dZv4dC1629ULLUNBMB4X7T62mJ/TlXJ/NDgwM1iXDTExMTlX
rNtzoz0v6dTELMAkAI1EBhW7C4Mx+EbArB3fja的时候N83UbGm18bJUowKwK7AP7B
BaC7CHNbih1B03n1I0Q4sWADAYQVRQeB4hj/EjMGfM9A1UECXMuHmzfG2HwGyD
VQQDEkA==<21:9LWldnrc1cJtcm9vc1DQ7/6CA0MGCRgSi3DQBAQJ/AasAa
ME9/CQWE1hbnll1U1vQDu70wb949nH7Fkmjvh2hL7GR16JV7/gHzmF27VZNP5I
VQ2/3INU5htj/QRdMeMlJ/raJX/72AmBAAgjYgewgkCw/vDVR0PfIA0DAGhHZM8G
AlBDwEwB/0QPMuBBg6h6WQV0R0BBBEYCPUYLUBfKND9gUonHnoSvHgO5kMIQG
BgfJVR8IEgYWy6QgQ6Bd+GWPW0dHA6Iy9jaXnjby10GO2b2hwb1ivQ2VydEYv
uc9oBcC92aW1c9uLWldnR1c3Jtcm9vc1DQ7S53cmwR/8DOtG7P2pGUG66lyyc
XG4LpC2NVLW4aJ2eHBUcLxDZXJ/0RW5yby2eXH7npbXZb24t2ZG52dGZvc1ibn290
LUNBnMnyDBagbkB5ggjYI3FQEEAwIBADANBgkqkhiO9w0BAQJFAANB0ICQe4W
YJal1G2qVLU4bDVMF06ELC2AMBgI4KL31xZ/2/03PjD7ctZBIAF4lktv9pCe61O
BoBcm2TeA+TQCg=
-----END CERTIFICATE-----
```

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

```
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# *
ssl-proxy(config)# ssl-proxy service client-proxy-sig-only client
ssl-proxy(config-ca-proxy)# virtual ipaddr 14.0.0.3 protocol tcp port 81
ssl-proxy(config)-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp 443
ssl-proxy(config)-ssl-proxy)# trusted-ca rootca
ssl-proxy(config)# authenticate verify signature-only
ssl-proxy(config)# inservice
```

This example shows how to configure server certificate authentication verifying all:

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(config-ca-trustpoint)# enrollment terminal
ssl-proxy(config-ca-trustpoint)#
```

Enter the base 64 encoded CA certificate. End with a blank line or the word "quit" on a line by itself.
-----BEGIN CERTIFICATE-----
MlIC1zCCAcGAgIBAgIQAQEJQQAAiEAiEAkDQYJKoZIhvcNAQcCAQ8wDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvc
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:

yes
Trustpoint CA certificate accepted.

% Certificate successfully imported

ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)#

This example shows how to configure server certificate authentication verifying all; the SSL module is configured as a client and sends its certificate to the SSL server if it is requested.

ssl-proxy(config)#
crypto ca trustpoint rootca
ssl-proxy(config-ca-trustpoint)# enrollment terminal
ssl-proxy(config-ca-trustpoint)#

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

-----BEGIN CERTIFICATE-----
MlIC1zCCAcGAgIBAgIQAQEJQQAAiEAiEAkDQYJKoZIhvcNAQcCAQ8wDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvcNAQcBAQcwDQYJKoZIhvc
-----END CERTIFICATE-----

ssl-proxy(config)#
crypto ca authenticate rootca

Enter the base 64 encoded certificate. End with a blank line or the word 'quit' on a line by itself.
Certificate Revocation List

A certificate revocation list (CRL) is a time-stamped list that identifies certificates that should no longer be trusted. Each revoked certificate is identified in a CRL by its certificate serial number. When a participating peer device uses a certificate, that device not only checks the certificate signature and validity but also checks that the certificate serial number is not on that CRL.

Note

Downloading and using CRLs are time-consuming and CPU-intensive operations.

This section describes how to download and configure CRLs:

- Downloading the CRL, page 3-53
- Configuring CRL Options, page 3-54
- Updating a CRL, page 3-54
- Entering X.500 CDP Information, page 3-55
- Entering a CRL Manually, page 3-55
- Displaying CRL Information, page 3-56
- Deleting a CRL, page 3-56
**Chapter 3      Configuring the SSL Services Module**

**Configuring Certificate Authentication**

**Downloading the CRL**

If the certificate being validated contains a CRL distribution point (CDP) extension field, the module uses the CDP as the download path. The SSL module supports three types of CDPs:

- **HTTP URL**
  
  For example, http://hostname/file.crl

- **X.500 distinguished name (DN)**
  
  For example, CN=CRL,O=cisco,C=us

- **Lightweight Directory Access Protocol (LDAP) URL**
  
  For example, ldap://hostname/CN=CRL,O=cisco,C=us

If the certificate does not have a CDP, the SSL module looks for a trustpoint that is associated with the certificate. If the SSL module finds one or more associated trustpoints, the SSL module uses the first configured trustpoint to determine the download path and protocol using the SCEP enrollment URL. If there is no SCEP enrollment URL, the validation fails.

---

**Note**

When the configuration contains multiple trustpoints using one common certificate chain, the CRL download information of only the first listed trustpoint is used. If this first trustpoint is not configured with a SCEP enrollment URL, the validation fails, regardless of the configuration of the other trustpoints.

---

**Note**

When using SCEP to request for a CRL, you need to associate a keypair with the trustpoint. You can assign any existing keypair for this purpose. The keypair is used for signing the CRL download request.

The SSL module does not perform a CRL lookup on the root certificate authority certificates because of the following reasons:

- Many root certificate authority certificates do not contain a CDP extension. If the certificate authority does not support SCEP for CRL request, the CRL download fails.

- CRLs are signed by the root certificate authority. If the root certificate authority has been revoked, its CRL will probably become invalid. All trustpoints associated with a revoked root certificate authority should be deleted from the database as soon as the revocation is made known.

If the download path is not known, or if the download operation fails, the peer certificate chain is rejected.

After the module downloads the CRL, the module checks to see if the serial number of the certificate appears on the CRL. If the serial number of the certificate being validated appears on the CRL, the peer certificate chain is rejected.

---

**Note**

One or more certificates in the peer certificate chain can fail the CRL lookup, even though some of the certificate authority certificates are trusted (for example, a certificate authority certificate was revoked after it was imported, or the CRL that was downloaded for this certificate authority certificate has expired and the attempt to download a updated CRL has failed).
Configuring CRL Options

There can be more than one trustpoint associated with a root or subordinate certificate authority certificate. During certificate authentication, any of these trustpoints can be selected to determine CRL configuration. In order to obtain consistent authentication results, all of these trustpoints need to bear the same CRL configuration.

By default, the SSL module always performs a CRL lookup if the trustpoint has been selected to validate a certificate. If the CRL is not in the database or has expired, the SSL module downloads a CRL and saves it to the database for later use. If the CRL download fails, the SSL module rejects the certificate being validated.

You can configure two options for CRL lookup:

- **Best-effort**
  - If the SSL module finds a CRL in the database and has not expired, then the SSL module performs a CRL lookup. If the SSL module does not find CRL, the SSL module attempts to download a CRL. However, if the CRL download fails, the SSL module accepts the certificate.

- **Optional**
  - If the SSL module finds a CRL in the database and has not expired, then the SSL module performs a CRL lookup. If the SSL module does not find CRL, the SSL module accepts the certificate. The SSL module makes no attempt to download a CRL.

To configure CRL options, perform this task:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>ssl-proxy(config)# crypto ca trustpoint trustpoint-label Declares the trustpoint that your module should use. Enabling this command puts you in ca-trustpoint configuration mode.</td>
</tr>
</tbody>
</table>

Updating a CRL

A CRL can be reused with subsequent certificates until the CRL expires.

If the specified NextUpdate time of the CRL is reached, the CRL is deleted. Enter the `show crypto ca timers` command to display the time remaining for the CRL.

If a CRL has not expired yet, but you suspect that the contents of the CRL are out of date, you can download the latest CRL immediately to replace the old CRL.

To request immediate download of the latest CRL, enter the following command:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-proxy(config)# crypto ca crl request trustpoint_label</td>
<td>Requests an updated CRL. This command replaces the currently stored CRL with the newest version of the CRL.</td>
</tr>
</tbody>
</table>
**Note**

Downloading a new CRL overwrites any existing version.

The following example shows how to download the CRL associated with the trustpoint “tp-root:"

```
ssl-proxy(config)# crypto ca crl request tp-root
```

**Entering X.500 CDP Information**

You can enter the hostname and port if the CDP is in X.500 DN format. The query takes the information in the following form: `ldap://hostname:[port]`

For example, if a certificate being validated has the following:

- the X.500 DN is configured with `CN=CRL,O=Cisco,C=US`
- the associated trustpoint is configured with `crl query ldap://10.1.1.1`

then the two parts are combined to form the complete URL as follows:
`ldap://10.1.1.1/CN=CRL,O=Cisco,C=US`

**Note**

Note that the trustpoint should be associated with the issuer certificate authority certificate of the certificate being validated. If there is no such trustpoint in the database, the complete URL cannot be formed, and CRL download cannot be performed.

To query the CRL with the X.500 URL, enter this command:

```
ssl-proxy(config)# crypto ca crl request tp-root
```

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ssl-proxy(ca-trustpoint)# crl query ldap://hostname:[port]</code></td>
<td>Allows you to enter the hostname and port if the CDP is in X.500 DN format.</td>
</tr>
</tbody>
</table>

**Entering a CRL Manually**

If the certificate authority server does not publish the CRL online (through HTTP, LDAP, or SCEP), you can get a hexdump of the CRL offline and enter it manually.

To manually enter the CRL, perform this task:

```
ssl-proxy(config)# crypto ca certificate chain tp
ssl-proxy(config-cert-chain)# crl
```

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ssl-proxy(config)# crypto ca certificate chain name</code></td>
<td>Enters certificate chain configuration mode; name specifies the name of the certificate authority.</td>
</tr>
<tr>
<td><code>ssl-proxy(config-cert-chain)# crl</code></td>
<td>Allows you to enter the revocation list issued by the certificate authority manually.</td>
</tr>
<tr>
<td><code>ssl-proxy(config-pubkey)# quit</code></td>
<td>Exits data entry mode.</td>
</tr>
<tr>
<td><code>ssl-proxy(config-cert-chain)# end</code></td>
<td>Exits certificate chain configuration mode.</td>
</tr>
</tbody>
</table>

The following example shows how to manually enter a CRL:

```
ssl-proxy(config)# crypto ca certificate chain tp
ssl-proxy(config-cert-chain)# crl
```

Enter the CRL in hexadecimal representation....
### Configuring Certificate Authentication

#### Chapter 3  Configuring the SSL Services Module

### Configuring Certificate Authentication

#### Displaying CRL Information

To display information about the CRLs, enter this command:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ssl-proxy(config)# show crypto ca crls</code></td>
<td>Displays the expiration date and time of each CRL in the database.</td>
</tr>
</tbody>
</table>

This example shows the expiration date and time of each CRL in the database:

```
ssl-proxy# show crypto ca crls
CRL Issuer Name:
  CN = test-root-CA, OU = lab, O = cisco, L = san jose, ST = california, C = US
  LastUpdate:19:08:45 UTC Dec 3 2003
  NextUpdate:20:13:45 UTC Dec 3 2003
  Retrieved from CRL Distribution Point:
    http://test-ca/CertEnroll/test-root-CA.crl
```

#### Deleting a CRL

**Note**
CRLs are deleted globally and not per trustpoint.

To delete a CRL, enter the `no crl` command for any certificate chain of a trustpoint.

This example shows how to delete the CRL:

```
ssl-proxy(config)# crypto ca certificate chain tp1
ssl-proxy(config-cert-chain)# no crl
```
Certificate Security Attribute-Based Access Control

The Certificate Security Attribute-Based Access Control feature adds fields to the certificate that allow specifying an access control list (ACL), to create a certificate-based ACL.

For information on configuring certificate security attribute-based access control, refer to Certificate Security Attribute-Based Access Control at this URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t15/ftcrtacl.htm