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

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).

Tip 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:

http://www.cisco.com/en/US/docs/ios/sec_secure_connectivity/configuration/guide/sec_pki_overview_ps6350_TSD_Products_Configuration_Guide_Chapter.html

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 Services 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 Services Module authenticates certificates that it receives from external devices when you configure the SSL Services Module as an SSL server and you configure the server proxy to authenticate the client certificate, or when you configure the SSL Services Module as an SSL client. The SSL Services 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. See the "Checking the Certificate Status" section on page 3-58 for information on configuring Online Certificate Status Protocol (OCSP) and certificate revocation lists (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 3-65 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 Services Module.

However, the SSL Services 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 Services 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 4-7 for information on configuring HTTP header insertion. See the "Client Certificate Authentication" section on page 3-51 for information on configuring client certificate authentication.

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

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 3-11 for details.
- If you are using an external PKI system, do the following:
  - Generate PKCS12 or PEM files.
  - Import this file to the module.

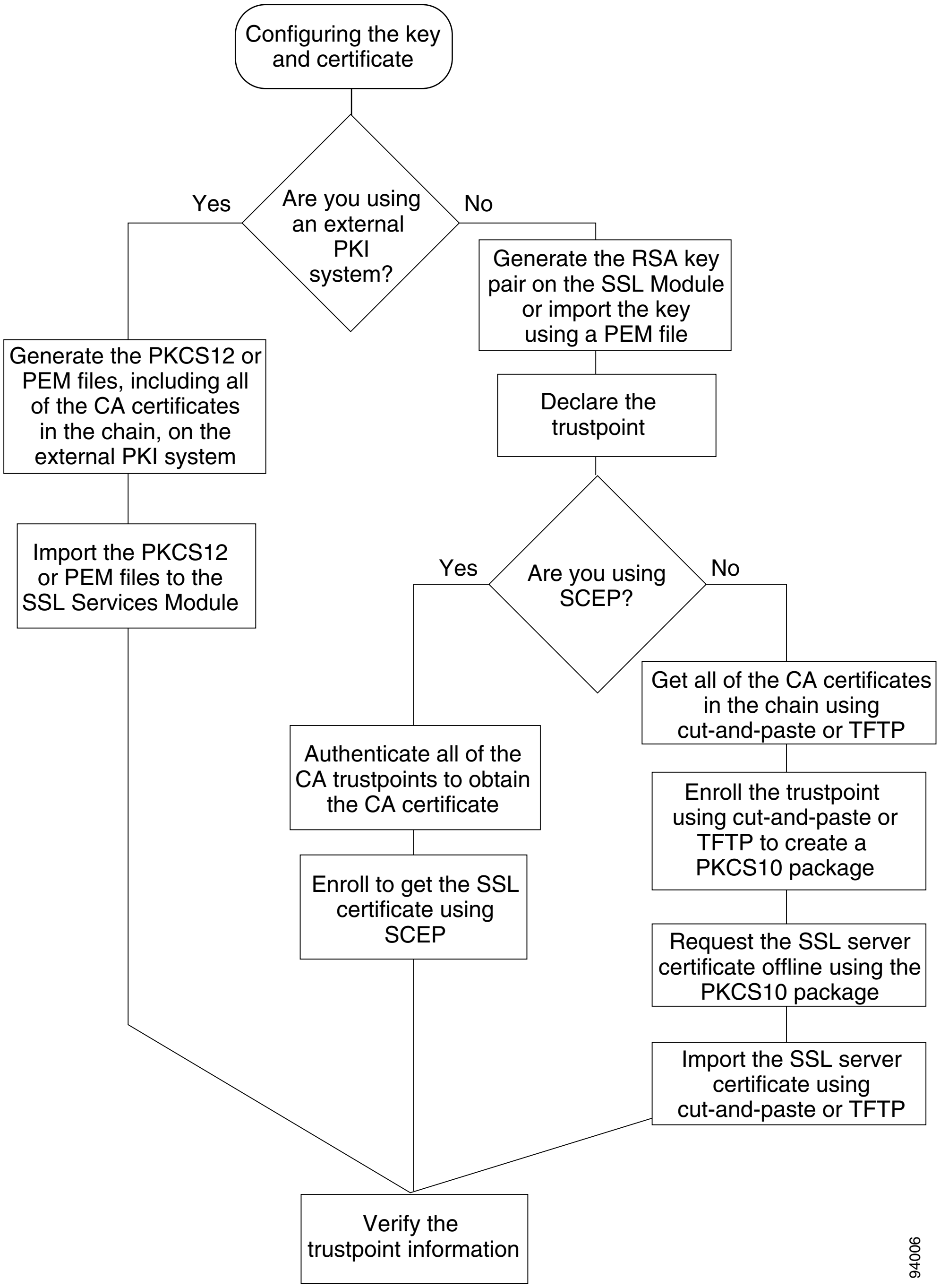
  See the "Importing and Exporting Key Pairs and Certificates" section on page 3-19 for details.

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


Configuring the key and certificate
Are you using an external PKI system?
Yes
No
Generate the PKCS12 or PEM files, including all of the CA certificates in the chain, on the external PKI system
Import the PKCS12 or PEM files to the SSL Services Module
Generate the RSA key pair on the SSL Module or import the key using a PEM file
Declare the trustpoint
Are you using SCEP?
Yes
No
Authenticate all of the CA trustpoints to obtain the CA certificate
Enroll to get the SSL certificate using SCEP
Get all of the CA certificates in the chain using cut-and-paste or TFTP
Enroll the trustpoint using cut-and-paste or TFTP to create a PKCS10 package
Request the SSL server certificate offline using the PKCS10 package
Import the SSL server certificate using cut-and-paste or TFTP
Verify the trustpoint information
94006

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 3-9

Generating RSA Key Pairs

Note 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.

Note 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:

Command	Purpose
ssl-proxy(config)# **crypto key generate rsa general-keys label** *key-label* [**exportable**[1]] [**modulus** *size*]	Generates RSA key pairs.

[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 kp1 exportable

The name for the keys will be: kp1

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].
```

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:

	Command	Purpose
Step 1	ssl-proxy(config)# **crypto pki trustpoint** *trustpoint-label*[1]	Declares the trustpoint that your module should use and enters 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[2].
Step 5	ssl-proxy(ca-trustpoint)# **revocation-check** *method1* [*method2*[*method3*]]	Checks the revocation status of a certificate. • **crl**—Certificate checking is performed by a CRL. This is the default option. • **none**—Certificate checking is ignored. • **ocsp**—Certificate checking is performed by an OCSP server. See the "Checking the Certificate Status" section on page 3-58 for information.
Step 6	ssl-proxy(ca-trustpoint)# **subject-name** *line*[3, 4]	(Optional) Configures the host name of the proxy service[5].
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:

```
ssl-proxy(config)# crypto pki trustpoint PROXY1
ssl-proxy(ca-trustpoint)# 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)# ocsp url http://myocspserver:81
ssl-proxy(ca-trustpoint)# revocation-check ocsp none
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:

Command	Purpose
`ssl-proxy(config)# crypto pki authenticate trustpoint-label`	Obtains the certificate that contains the public key of the certificate authority. Enter the same *trustpoint_label* that you entered when declaring the trustpoint.

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

```
ssl-proxy(config)# crypto pki 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:

Command	Purpose
ssl-proxy(config)# **crypto pki enroll** *trustpoint-label*[1]	Requests a certificate for the trustpoint.

[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 pki 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:

```
ssl-proxy(config)# crypto pki 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 pki 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 3-28 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

```
ssl-proxy(onfig)# 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

```
ssl-proxy(config)# crypto pki trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.1
ssl-proxy(ca-trustpoint)#
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.2
ssl-proxy(ca-trustpoint)#
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki 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 02050D480809
 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 pki 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 pki 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 pki 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 pki 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 pki 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 you have configured the **revocation-check crl** command, and 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 **revocation-check none** command for all trustpoints that are linked to the import process. Enter the **show crypto pki 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 **revocation-check none** 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 **revocation-check none** command for all the trustpoints associated with root CA. Similarly, when you import the subordinate CA2 certificate, enter the **revocation-check none** 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 pki trustpoint tftp_example
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.2/win2k
ssl-proxy(ca-trustpoint)# rsakeypair pair3
ssl-proxy(ca-trustpoint)# exit
```

2. Request a certificate for the trustpoint:

```
ssl-proxy(config)# crypto pki 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 pki 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:

```
ssl-proxy(config)# crypto pki 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:

```
ssl-proxy(config)# crypto pki 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]

ssl-proxy(config)#
*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:

```
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:

```
ssl-proxy(config)# crypto pki 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:

```
ssl-proxy(config)# crypto pki 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
A1UEAxMDYWJjMTMwDwYDVQQFEwhCMEZGRjIyRTAgBgkqhkiG9w0BCQIWE3NzbC1w
cm94eS5jaXNjby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALt7O6tt
30lBVVK1qAE/agsuzIaa15YZft3bDb9t3pPncKh0ivBTgVKpJiLPWGZPjdbtejxQ
tYSF77R1pmhK0WSKPuu7fJPYr/Cbo80OUzkRAgMBAAGgITAfBgkqhkiG9w0BCQ4x
EjAQMA4GA1UdDwEB/wQEAwIFoDANBgkqhkiG9w0BAQQFAAOBgQC2GIX06/hihXHA
DA5sOpxgLsO1rMP8PF4bZDdlpWLVBSOrp4S1L7hH9P2NY9rgZAJhDTRfGGm179JY
GOtUuCyPYPkpb0S5VGTUrHvvUWekleKq2d91kfgbkRmJmHBaB2Ev5DNBcV11SIMX
RULG7oUafU6sxnDWqbMseToF4WrLPg==

---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 pki 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
MBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50ZXJuZXQgV2lkZ2l0cyBQ
dHkgTHRkMQswCQYDVQQDEwJjYTAeFw0wMzA2MjYyMjM4MDlaFw0wODEyMTYyMjM4
MDlaMFIxCzAJBgNVBAYTAkFVMRMwEQYDVQQIEwpTb21lLVN0YXRlMSEwHwYDVQQK
ExhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQxCzAJBgNVBAMTAmNhMIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQCcG9ObqOLmf0cASkF48jz8X7ZQxT1H68OQKNC3
ks95vkGbOAa/1/R4ACQ3s9iPkcGQVqi4Dv8/iNG/1mQo8HBwtR9VgG0l8IGBbuiZ
dlarYnQHUz6Bm/HzE1RXVOY/VmyPOVevYy8/cYhwx/xOE9BYQOyP15Chi8nhIS5F
+WWoHQIDAQABo4GsMIGpMB0GA1UdDgQWBBS4Y+/lSXKDrw5N5m/tgCzu/W81PDB6
BgNVHSMEczBxgBS4Y+/lSXKDrw5N5m/tgCzu/W81PKFWpFQwUjELMAkGA1UEBhMC
QVUxEzARBgNVBAgTClNvbWUtU3RhdGUxITAfBgNVBAoTGEludGVybmV0IFdpZGdp
dHMgUHR5IEx0ZDELMAkGA1UEAxMCY2GCAQAwDAYDVR0TBAUwAwEB/zANBgkqhkiG
9w0BAQQFAAOBgQB/rPdLFVuycbaJQucdFQG7kl/XBNI7aY3IL3Lkeumt/nXD+eCn
RpYE5WWY8X1Aizqnj4bqFdqPqYdD7Lg8viwqm2tQmU6zCsdaKhL1J7FCWbfs2+Z5
oNV2Vsqx0Ftnf8en/+HtyS2AdXHreThfgkXz3euXD0ISMFVKRy81o4EdzA==
-----END 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

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 pki import CSR-TP certificate
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com

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

-----BEGIN CERTIFICATE-----
MIIB7TCCAVYCAQQwDQYJKoZIhvcNAQEEBQAwUjELMAkGA1UEBhMCQVUxEzARBgNV
BAgTClNvbWUtU3RhdGUxITAfBgNVBAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0
ZDELMAkGA1UEAxMCY2EwHhcNMDMxMTIwMDAxMzE2WhcNMDQxMTE5MDAxMzE2WjAs
MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMQwwCgYDVQQDEwNhYmMwgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALt7O6tt30lBVVK1qAE/agsuzIaa15YZ
ft3bDb9t3pPncKh0ivBTgVKpJiLPWGZPjdbtejxQksuSY589V+GMDrO9B4Sxn+5N
p2bQmd745NvI4gorNRvXcdjmE+/SzE+bBSBcKAwNtYSF77R1pmhK0WSKPuu7fJPY
r/Cbo80OUzkRAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAjqJ9378P6Gz69Ykplw06
Powp+2rbe2iFBrE1xE09BL6G6vzcBQgb5W4uwqxe7SIHrHsS0/7Be3zeJnlOseWx
/KVj7I02iPgrwUa9DLavwrTyaa0KtTpti/i5nIwTNh5xkp2bBJQikD4TEK7HAvXf
HQ9SyB3YZJk/Bjp6/eFHEfU=
-----END CERTIFICATE-----

% Router Certificate successfully imported

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

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 pki 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 pki 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:B0FFF22E
% 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 pki certificate' command will also show the fingerprint.

ssl-proxy(config)# Fingerprint: 19B07392 319B2ACF F8FABE5C 52798971

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

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 pki trustpoint test-1tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-1tier
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki 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 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.

ssl-proxy(config)# crypto pki trustpoint test-2tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-2tier
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki 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 pki 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 pki 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:

```
ssl-proxy(config)# crypto pki trustpoint tp-proxy1
ssl-proxy(ca-trustpoint)# enrollment ter
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:

```
ssl-proxy(config)# crypto pki 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:

MIIBnDCCAQUCAQAwOzENMAsGA1UEAxMEdGVzdDEqMA8GA1UEBRMIQjBGRkYxNEQw
FwYJKoZIhvcNAQkCFgpzc2wtcHJveHkuMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQDFx1ol9IXoAx4fyUhaXH6s4p5t9soIZ1gvLtVX6Fp6zfuX47os5TGJH/IX
zV9B4e5Kv+wlMD0AvTh+/tvyAP3TMpCdpHYosd2VaTIgExpHf4M5Ruh8IebVKV25
rraIpNiS0PvPLFcrw4UfJVNpsc2XBxBhpT+FS9y67LqlhfSN4wIDAQABoCEwHwYJ
KoZIhvcNAQkOMRIwEDAOBgNVHQ8BAf8EBAMCBaAwDQYJKoZIhvcNAQEEBQADgYEA
kOIjd1KNJdKLMf33YELRd3MW/ujJIuiT1J8RYVbw1eE8JQf68TTdKiYqzQcoMgsp
ez3vSPxXFZ/c6naXdVyrTikTX3GZ1mu+UOvV6/Jaf5QcXa9tAi3fgyguV7jQMPjk
Qj2GrwhXjcqZGOMBh6Kq6s5UPsIDgrL036I42B6B3EQ=

---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 truspoints for root certificate authority and subordinate 1 certificate authority.

   Note We will use **tp-proxy1** to import the subordinate 2 certificate authority certificate.

```
ssl-proxy(config)# crypto pki trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
```

   b. Import the root certificate authority certificate:

```
ssl-proxy(config)# crypto pki 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-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----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. Import subordinate 1 certificate authority certificate:

```
ssl-proxy(config)# crypto pki 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
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMzIyMDQyMVoXDTA0MTExMzIyMTQyMVow
dTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdz
aW1wc29uLWRldnRlc3Qtc3ViMS1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQDc
```

```
vV48nC2uukoSyGJ/GymCIEXZzMSzpbkYS7eWPaZYyiJDhCIKuUsMgFDRNfMQmUSA
rcWmPizFZc9PFumDa03vAgMBAAGjggJpMIICZTAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQUWaaNN2U14BaBoU9mY+ncuHpP920wCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wga4GA1UdIwSBpjCBo4AUJgYtQFMo130SBSiceehK9seDRrGh
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCEGnVMc1P4ve4Q5mUWCdWwXAwgZcG
A1UdHwSBjzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWw4ajZvaHBuci9DZXJ0RW5y
b2xsL3NpbXBzb24tZGV2dGVzdC1yb290LUNBLmNybDBFoEOgQYY/ZmlsZTovL1xc
Y2lzY28tbDhqNm9ocG5yXENlcnRFbnJvbGxcc2ltcHNvbi1kZXZ0ZXN0LXJvb3Qt
Q0EuY3JsMIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Np
c2NvLWw4ajZvaHBuci9DZXJ0RW5yb2xsL2Npc2NvLWw4ajZvaHBucl9zaW1wc29u
LWRldnRlc3Qtcm9vdC1DQS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2Nv
LWw4ajZvaHBuclxDZXJ0RW5yb2xsXGNpc2NvLWw4ajZvaHBucl9zaW1wc29uLWRl
dnRlc3Qtcm9vdC1DQS5jcnQwDQYJKoZIhvcNAQEFBQADQQA6kAV3Jx/BOr2hlSp9
ER36ZkDJNIW93gNt2MkpcA07RmcrHln6q5RJ9WbvTxFnONdgpsag1EcOwn97XErH
Z2ow
-----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 pki 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-----
MIIESTCCA/OgAwIBAgIKHyiFxAAAAAAABjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIxLWNhMB4XDTAzMTExMzIyMjI1MloXDTA0MTExMzIyMTQyMVow
dTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdz
aW1wc29uLWRldnRlc3Qtc3ViMi1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC7
ChZc0NYLBHf1sr/3Z4y6w5WoeioIpCOCSydhnbd5wnwuethoyStVt9lr6i61jWKl
d68Z8EoTg71daiV/WR/HAgMBAAGjggJjMIICXzAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQU6FmJopqqzpbFMj6TaB2/wjlWlqEwCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wgagGA1UdIwSBoDCBnYAUWaaNN2U14BaBoU9mY+ncuHpP922h
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCCho9HAcAAAAAAA4wgZcGA1UdHwSB
jzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xsL3Np
bXBzb24tZGV2dGVzdC1zdWIxLWNhLmNybDBFoEOgQYY/ZmlsZTovL1xcY2lzY28t
ZzI1dWE2bzRlXENlcnRFbnJvbGxcc2ltcHNvbi1kZXZ0ZXN0LXN1YjEtY2EuY3Js
MIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Npc2NvLWcy
NXVhNm80ZS9DZXJ0RW5yb2xsL2Npc2NvLWcyNXVhNm80ZV9zaW1wc29uLWRldnRl
c3Qtc3ViMS1jYS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2NvLWcyNXVh
Nm80ZVxDZXJ0RW5yb2xsXGNpc2NvLWcyNXVhNm80ZV9zaW1wc29uLWRldnRlc3Qt
c3ViMS1jYS5jcnQwDQYJKoZIhvcNAQEFBQADQQCieB8rvVCqVF2cFw9/v51jGn7L
Q6pUGT3bMRbOrgQKytTz/Yx09156nYZHrvVuLzmzz5CriI2saVx+q1Tarwil
-----END CERTIFICATE-----

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

e. Import the server certificate:

```
ssl-proxy(config)# crypto pki import tp-proxy1 certificate
% The fully-qualified domain name in the certificate will be:ssl-proxy.

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

-----BEGIN CERTIFICATE-----
MIIENTCCA9+gAwIBAgIKLmibDwAAAAAACDANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIyLWNhMB4XDTAzMTExOTIzNDUzNVoXDTA0MTExMzIyMTQyMVow
PTERMA8GA1UEBRMIQjBGRkYxNEQxGTAXBgkqhkiG9w0BCQITCnNzbC1wcm94eS4x
DTALBgNVBAMTBHRlc3QwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMXHWiX0
hegDHh/JSFpcfqzinm32yghnWC8u1VfoWnrN+5fjuizlMYkf8hfNX0Hh7kq/7CUw
Af8EBAMCBaAwHQYDVR0OBBYEFCXlzcYHyo1PNbfubnivi8d2VO22MIGoBgNVHSME
gaAwgZ2AFOhZiaKaqs6WxTI+k2gdv8I5VpahoXmkdzB1MQswCQYDVQQGEwJVUzET
MBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2UxDjAMBgNVBAoT
BWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24tZGV2dGVzdC1z
dWIxLWNhggofKIXEAAAAAAAGMIGXBgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9j
aXNjby1vam14Y25jenYvQ2VydEVucm9sbC9zaW1wc29uLWRldnRlc3Qtc3ViMi1j
YS5jcmwwRaBDoEGGP2ZpbGU6Ly9cXGNpc2NvLW9qbXhjbmN6dlxDZXJ0RW5yb2xs
XHNpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNybDCByAYIKwYBBQUHAQEEgbswgbgw
WQYIKwYBBQUHMAKGTWh0dHA6Ly9jaXNjby1vam14Y25jenYvQ2VydEVucm9sbC9j
aXNjby1vam14Y25jenZfc2ltcHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3J0MFsGCCsG
AQUFBzAChk9maWxlOi8vXFxjaXNjby1vam14Y25jenZcQ2VydEVucm9sbFxjaXNj
by1vam14Y25jenZfc2ltcHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3J0MA0GCSqGSIb3
DQEBBQUAA0EAtbxmUBOxZ/hcrCc3hY7pa6q/LmLonXSL8cjAbV2I7A5QGYaNi5k9
8FlEz1WOxW0J2C3/YsvIf4dYpsQWdKRJbQ==
-----END CERTIFICATE-----

% Router Certificate successfully imported

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

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 you have configured the **revocation-check crl** command, and 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 **revocation-check none** command for all trustpoints that are linked to the import process. Enter the **show crypto pki 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 **revocation-check none** 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 **revocation-check none** command for all the trustpoints associated with root CA. Similarly, when you import the subordinate CA2 certificate, enter the **revocation-check none** 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:

Command	Purpose
`ssl-proxy(config)#` **`crypto pki`** `{`**`import`** `|` **`export`**`}` *`trustpoint_label`* **`pkcs12`** `{`**`scp:`**`|` **`ftp:`**`|` **`nvram:`**`|` **`rcp:`**`|` **`tftp:`**`}` `[`*`pkcs12_filename`*[1]`]` *`pass_phrase`*[2]	Imports or exports a PKCS12 file. **Note** You do not need to configure a trustpoint before importing the PKCS12 file. Importing keys and certificates from a PKCS12 file creates the trustpoint automatically, if it does not already exist.

[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 pki 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 12:30:00.531:%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 pki 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.
ssl-proxy(config)#
```

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

```
ssl-proxy(config)# crypto pki 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]
ssl-proxy(config)#
```

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

```
ssl-proxy(config)# crypto pki 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.
ssl-proxy(config)#
```

After you import the PKCS12 file, see the "Verifying Certificates and Trustpoints" section on page 3-28 to verify the certificate and trustpoint information.

Importing and Exporting PEM Files

Note The **crypto pki 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:

Command	Purpose
ssl-proxy(config)# **crypto pki import** *trustpoint_label* **pem** [**exportable**] {**terminal** | **url** {**scp:**| **ftp:**| **nvram:**| **rcp:**| **tftp:**} | **usage-keys**} *pass_phrase*[1,2]	Imports PEM files. **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.
ssl-proxy(config)# **crypto pki export** *trustpoint_label* **pem** {**terminal** | **url** {**scp:**| **ftp:**| **nvram:**| **rcp:**| **tftp:**} [**des** | **3des**] *pass_phrase*[1,2]	Exports PEM files. **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:

Note The TP5.ca, TP5.prv, and TP5.crt files should be present on the server.

```
ssl-proxy(config)# crypto pki import TP5 pem url tftp://10.1.1.1/TP5 password
% Importing CA certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [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]?
Destination filename [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]?
Destination filename [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
ssl-proxy#
*Apr 11 15:11:29.901: %SYS-5-CONFIG_I: Configured from console by console
```

This example shows how to export PEM files using TFTP:

```
ssl-proxy(config)# crypto pki export TP5 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!
ssl-proxy(config)#
```

After you import the PEM files, see the "Verifying Certificates and Trustpoints" section on page 3-28 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:

```
ssl-proxy(config)# crypto pki trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki 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-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----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
```

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

```
ssl-proxy(config)# crypto pki trustpoint 3tier-subca1
ssl-proxy(ca-trustpoint)# enroll terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki 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
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMzIyMDQyMVoXDTA0MTExMzIyMTQyMVow
dTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdz
aW1wc29uLWRldnRlc3Qtc3ViMS1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQDc
vV48nC2uukoSyGJ/GymCIEXZzMSzpbkYS7eWPaZYyiJDhCIKuUsMgFDRNfMQmUSA
rcWmPizFZc9PFumDa03vAgMBAAGjggJpMIICZTAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQUWaaNN2U14BaBoU9mY+ncuHpP920wCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wga4GA1UdIwSBpjCBo4AUJgYtQFMo130SBSiceehK9seDRrGh
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCEGnVMc1P4ve4Q5mUWCdWwXAwgZcG
A1UdHwSBjzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWw4ajZvaHBuci9DZXJ0RW5y
b2xsL3NpbXBzb24tZGV2dGVzdC1yb290LUNBLmNybDBFoEOgQYY/ZmlsZTovL1xc
Y2lzY28tbDhqNm9ocG5yXENlcnRFbnJvbGxcc2ltcHNvbi1kZXZ0ZXN0LXJvb3Qt
Q0EuY3JsMIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Np
c2NvLWw4ajZvaHBuci9DZXJ0RW5yb2xsL2Npc2NvLWw4ajZvaHBucl9zaW1wc29u
LWRldnRlc3Qtcm9vdC1DQS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2Nv
LWw4ajZvaHBuclxDZXJ0RW5yb2xsXGNpc2NvLWw4ajZvaHBucl9zaW1wc29uLWRl
dnRlc3Qtcm9vdC1DQS5jcnQwDQYJKoZIhvcNAQEFBQADQQA6kAV3Jx/BOr2hlSp9
ER36ZkDJNIW93gNt2MkpcA07RmcrHln6q5RJ9WbvTxFnONdgpsag1EcOwn97XErH
Z2ow
-----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
```

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 pki import tp-proxy1 pem terminal cisco
% Enter PEM-formatted CA certificate.
% End with a blank line or "quit" on a line by itself.
-----BEGIN CERTIFICATE-----
MIIESTCCA/OgAwIBAgIKHyiFxAAAAAAABjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIxLWNhMB4XDTAzMTExMzIyMjI1MloXDTA0MTExMzIyMTQyMVow
dTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdz
aW1wc29uLWRldnRlc3Qtc3ViMi1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC7
ChZc0NYLBHf1sr/3Z4y6w5WoeioIpCOCSydhnbd5wnwuethoyStVt9lr6i61jWKl
d68Z8EoTg71daiV/WR/HAgMBAAGjggJjMIICXzAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQU6FmJopqqzpbFMj6TaB2/wjlWlqEwCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wgagGA1UdIwSBoDCBnYAUWaaNN2U14BaBoU9mY+ncuHpP922h
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCCho9HAcAAAAAAA4wgZcGA1UdHwSB
jzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xsL3Np
bXBzb24tZGV2dGVzdC1zdWIxLWNhLmNybDBFoEOgQYY/ZmlsZTovL1xcY2lzY28t
ZzI1dWE2bzRlXENlcnRFbnJvbGxcc2ltcHNvbi1kZXZ0ZXN0LXN1YjEtY2EuY3Js
MIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Npc2NvLWcy
NXVhNm80ZS9DZXJ0RW5yb2xsL2Npc2NvLWcyNXVhNm80ZV9zaW1wc29uLWRldnRl
c3Qtc3ViMS1jYS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2NvLWcyNXVh
Nm80ZVxDZXJ0RW5yb2xsXGNpc2NvLWcyNXVhNm80ZV9zaW1wc29uLWRldnRlc3Qt
c3ViMS1jYS5jcnQwDQYJKoZIhvcNAQEFBQADQQCieB8rvVCqVF2cFw9/v51jGn7L
Q6pUGT3bMRbOrgQKytTz/Yx09156nYZHrvVuLzmzz5CriI2saVx+q1Tarwil
-----END CERTIFICATE-----

% 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

gQb9JMplIE5AEdhumLuBFWT53k+L/EGLhFfQn/roPlEOiIGEB6y3DeYNN/xZSiy3
JOHN0kh8Wjw3pshrdNVcoQj2X7BPI+YOipok40WOk5J/+dnRLwMjv+rl0tr+LcCk
nBdR8zIOkOJObULLUOXFBM7oB3Dsk4Y3FBv8EAR3AdQiZjevau4FIyQn+JfVZy+J
wctmvZnX0c0fevPsgID4dCPkeY6+I0DkxMyRiuyn+wIrJw1xVA2VIOrRJojBNlRu
6/APef8JwpfnNcgpcLYt/4Q+3Yjl9EfRLjgiL6eSRki/6K5lrV3eKbwOTyjvXq5h
G0Q6dtNEoIvOg1Vad0CXeL+TxJ4ySq4E63OxIHkclDBsusGoUGLoZ+OtaxApAZ+5
WbKqR+ND1LlPmS8/ZL9LMPhUh9eOqZJjJTe6NbxY7jeNHjAmpP7/WpB2f2kV/LZg
n2AV4GALBZtqXtreGiayZzXpEA5J00lbzRZWf9JHA1diz/unW00/GH9LvCqA9O15
YJGCrRMI9US7MWm8kIkiJqNgLtbPad5cOaieQe+Kncgcm18Hc7pfhDwXGG4RS40x
TSV/kIR4Gi7h8Lu71wZKTaWYHBPTUyTIpNsFUEdvItHXOSBw2LWNWzdYgpGoMT/t
ryuu0lAC9YdBalAxY0DaqqpuXKzxfiw5QDbqZWVq3qAxXfLAtTgu/gFCuFQvbBGl
87H1C+nOQUq2nkpMpHZLsl3V0w/2yqg+q6rUydANFF+a5vRaLgX/PGms92ZkZUdP
Z5qeKJmoURSlMYxDuhQDl93RYxXJxOYIYrCrI/QaBpIH6QvUH60wWA==
-----END RSA PRIVATE KEY-----
quit
% Enter PEM-formatted certificate.
% End with a blank line or "quit" on a line by itself.
```

```
-----BEGIN CERTIFICATE-----
MIIEXTCCBAegAwIBAgIKTJOcWgAAAAAACTANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIyLWNhMB4XDTAzMTEyNTIwMjExMFoXDTA0MTExMzIyMTQyMVow
PjERMA8GA1UEBRMIQjBGRkYyMkUxKTAnBgkqhkiG9w0BCQITGnNpbXBzb24tNjUw
OS1zdGUuY2lzY28uY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCkhRKm
38hSF7l0WXlYm8ixs2Hz/yjNw7tchtRPIp0qCTJKW00gzZpp8dqaNi3s2GVVWb+t
Cgsol0MZLIkyoj/9vT9MC7Zo3LOxYy9kD+6M9peUWMT4JLSD4Exzxsd87JpP1bo0
o8WhYjvMor/bL30sW8ly2RH2vppEMn9eLEN0vwIDAQABo4ICajCCAmYwCwYDVR0P
BAQDAgWgMB0GA1UdDgQWBBSx6uQ2sARlcjzhBSiMu7xeu1n6AjCBqAYDVR0jBIGg
MIGdgBToWYmimqrOlsUyPpNoHb/COVaWoaF5pHcwdTELMAkGA1UEBhMCVVMxEzAR
BgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVj
aXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdzaW1wc29uLWRldnRlc3Qtc3Vi
MS1jYYIKHyiFxAAAAAAABjAoBgNVHREBAf8EHjAcghpzaW1wc29uLTY1MDktc3Rl
LmNpc2NvLmNvbTCBlwYDVR0fBIGPMIGMMEOgQaA/hj1odHRwOi8vY2lzY28tb2pt
eGNuY3p2L0NlcnRFbnJvbGwvc2ltcHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3JsMEWg
Q6BBhj9maWxlOi8vXFxjaXNjby1vam14Y25jenZcQ2VydEVucm9sbFxzaW1wc29u
LWRldnRlc3Qtc3ViMi1jYS5jcmwwgcgGCCsGAQUFBwEBBIG7MIG4MFkGCCsGAQUF
BzAChk1odHRwOi8vY2lzY28tb2pteGNuY3p2L0NlcnRFbnJvbGwvY2lzY28tb2pt
eGNuY3p2X3NpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNydDBbBggrBgEFBQcwAoZP
ZmlsZTovL1xcY2lzY28tb2pteGNuY3p2XENlcnRFbnJvbGxcY2lzY28tb2pteGNu
Y3p2X3NpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNydDANBgkqhkiG9w0BAQUFAANB
ABFh7XeLwvfBtjAR+e5OaUH5KTGJDbeJppOmMFXnFakpgWop9Qg4cHRCQq7V0pAW
iA6VtJOmpYgEIVNTAzAAHR4=
-----END CERTIFICATE-----

% 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
ssl-proxy#
```

4. Display the certificate information (optional):

```
ssl-proxy# show crypto pki certificates tp-proxy1
Certificate
  Status:Available
  Certificate Serial Number:04A0147B00000000010E
  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:6D1E6B0F000000000007
  Certificate Usage:Signature
  Issuer:
    CN = subtest
     C = US
Subject:
    CN = sub3ca
     C = US
CRL Distribution Point:
    http://sample.cisco.com/subtest.crl
  Validity Date:
    start date:22:22:52 UTC Mar 28 2003
    end   date:21:05:17 UTC Dec 12 2003
  Associated Trustpoints:tp-proxy1

ssl-proxy# show crypto pki certificates 3tier-subca1
CA Certificate
  Status:Available
  Certificate Serial Number:29A47DEF0000000004E9
  Certificate Usage:Signature
  Issuer:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
     C = US
Subject:
    CN = subtest
     C = US
CRL Distribution Point:
    http://sample.cisco.com/6ebf9b3e-9a6d-4400-893c-dd85dcfe911b.crl
  Validity Date:
    start date:20:55:17 UTC Dec 12 2002
    end   date:21:05:17 UTC Dec 12 2003
  Associated Trustpoints:3tier-sub1

ssl-proxy# show crypto pki certificates 3tier-root
CA Certificate
  Status:Available
  Certificate Serial Number:7FD5B209B5C2448C47F77F140625D265
  Certificate Usage:Signature
  Issuer:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
     C = US
Subject:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
     C = US
CRL Distribution Point:
    http://sample.cisco.com/6ebf9b3e-9a6d-4400-893c-dd85dcfe911b.crl
  Validity Date:
    start date:00:05:32 UTC Jun 13 2002
    end   date:00:11:58 UTC Jun 13 2004
  Associated Trustpoints:3tier-root
```

Verifying Certificates and Trustpoints

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

	Command	Purpose
Step 1	`ssl-proxy(ca-trustpoint)# show crypto pki certificates [trustpoint_label]`	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.
Step 2	`ssl-proxy(ca-trustpoint)# show crypto pki trustpoints [trustpoint_label]`	Displays information about all trustpoints or the specified trustpoint.

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 Services 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:

Command	Purpose
`ssl-proxy# copy [/erase] system:running-config nvram:startup-config`	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. **Note** 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. **Caution** 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.

Note 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:

	Command	Purpose
Step 1	`ssl-proxy# show startup-config`	Displays the startup configuration.
Step 2	`ssl-proxy# directory nvram:`	Displays the names and sizes of the files in NVRAM.

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:

Command	Purpose
`ssl-proxy# erase nvram:`	Erases the startup configuration and the key pairs.
`ssl-proxy# erase /all nvram:`	Erases the startup configuration, the key pairs, the certificates, and all other files from the NVRAM.

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.

See Importing and Exporting a PKCS12 File, page 3-20.

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 3-31
- Viewing Keys and Certificates, page 3-32
- Deleting Certificates from the Configuration, page 3-32

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:

Command	Purpose
`ssl-proxy(config)# crypto key zeroize rsa [key-label]`	Deletes all RSA key pairs, or the specified key pair. **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 pki 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 3-32.

Viewing Keys and Certificates

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

Command	Purpose
`ssl-proxy#` **`show crypto key mypubkey rsa`**	Displays RSA public keys for the module.
`ssl-proxy#` **`show crypto pki certificates`** `[`*`trustpoint_label`*`]`	Displays information about the certificate, the certificate authority certificate, and any registration authority certificates.
`ssl-proxy#` **`show running-config`** `[`**`brief`**`]`	Displays the public keys and the certificate chains. If the *brief* option is specified, the hex dump of each certificate is not displayed.
`ssl-proxy#` **`show ssl-proxy service`** *`proxy-name`*	Displays the key pair and the serial number of the certificate chain used for a specified proxy service. **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:

Command	Purpose
`ssl-proxy(config)#` **`no crypto pki trustpoint`** *`trustpoint-label`*	Deletes the certificate and trustpoint.

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:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service s2
ssl-proxy(config-ctx-ssl-proxy)# virtual ip 10.1.1.2 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ip 20.0.0.3 p tcp p 80
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general trustpoint tp-1
ssl-proxy(config-ctx-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-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#
```

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

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
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service s2
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general trustpoint tp-2
ssl-proxy(config-ctx-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:70FCBFEC000100000D65
    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:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service s2
ssl-proxy(config-ctx-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:1DFBB1FD000100000D48
    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.

```
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:313AD6510D25ABAE4626E96305511AC4
  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.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# crypto pki 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 pki certificate' command will also show the fingerprint.

    Fingerprint: 6518C579 A0498063 C5795057 A6170 075

ssl-proxy(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:313AD6510D25ABAE4626E96305511AC4
  Obsolete certificate chain in use for old connections:
    Server Certificate:
       Key Label:k2
       Serial Number:1DFBB1FD000100000D48
    Root CA Certificate:
      Serial Number:313AD6510D25ABAE4626E96305511AC4
  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.

```
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:313AD6510D25ABAE4626E96305511AC4
  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:

	Command	Purpose
Step 1	`ssl-proxy(config)# `**`crypto pki trustpoint`** *`trustpoint-label`*	Declares the trustpoint.
Step 2	`ssl-proxy(ca-trustpoint)# `**`auto-enroll`**` {renewal_`*`percent`*`|`**`regenerate`**`}`	Enables automatic renewal and enrollment for the specified trustpoint. **Note** Valid values for *renewal_percent* are 0 (enroll within 1 minute) through 100. **Note** The **regenerate** keyword generates a new key for the certificate even if a named key already exists.
Step 3	`ssl-proxy# `**`show crypto pki timers`**	Displays the time remaining before each timer expires.

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 pki trustpoint tk21
ssl-proxy(ca-trustpoint)# auto-enroll 90
ssl-proxy(ca-trustpoint)# end
ssl-proxy# show crypto pki 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:

	Command	Purpose
Step 1	`ssl-proxy(config)# `**`ssl-proxy pki history`**	Enables key and certificate history.
Step 2	`ssl-proxy# `**`show ssl-proxy certificate-history`**` [`**`service`** *`proxy_service`*`]`	Displays key and certificate history records for all services or the specified service.

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 Services Module to cache authenticated server and client (peer) certificates. Caching authenticated peer certificates saves time by not requiring the SSL Services Module to authenticate the same certificate again. The SSL Services 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 Services Module to incorrectly accept a certificate that should be denied. For instance, the SSL Services 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 Services 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 Services 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 Services 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 time, 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:

Command	Purpose
ssl-proxy(config)# **ssl-proxy pki cache size** *size* **timeout** *minutes*	Configures the certificate cache parameters. The default value for *size* is 0 (disabled); valid values are 0 through 5000 entries. The default value for *minutes* is 15 minutes; valid values are 1 through 600 minutes. To clear the cache, change the cache size or the timeout value.

Configuring Certificate Expiration Warning

You can configure the SSL Services 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 Services 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.

Note The SSL Services 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 Services 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 4-18.

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

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

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 CApool is going
to expire or has expired at this time:22:19:38 UTC Mar 4 2004, CA pool:pool2,
trustpoint:tp1-root.
```

Protected Private Key Storage

Digital signatures are a common method used to authenticate one device to another device. Certificates provide a key distribution mechanism that is required by digital signatures and public key cryptography. To use digital signatures, private information (the private key) must be stored on the device that is providing the signature. The stored private information may aid an attacker who steals the hardware device that contains the private key; for example, an attack may be able to cause the router to initiate a secure connection to another site by using the RSA private keys stored in the router.

The Protected Private Key Storage feature allows you to encrypt the private key that is stored in NVRAM by using a passphrase. In addition, you can also “lock” the private key, which blocks new connection attempts from a running router and protects the key in the router if the router is stolen by an attempted attacker.

Prerequisites

Before you can encrypt or lock a private key, you should perform the following tasks:

- Generate the RSA keys. (We recommend that you do not generate the RSA keys as exportable; however, doing so does not have an affect on this feature.)
- Optionally, you can authenticate and enroll each router with the certification authority (CA) server.

Note The RSA keys must be unlocked while enrolling the CA. The keys can be locked while authenticating the router with the CA because the private key of the router is not used during authentication.

Passphrase Limitations

If a passphrase is lost, you must regenerate the key, enroll with the CA server again, and obtain a new certificate. A lost passphrase cannot be recovered.

If you want to change a passphrase, you must decrypt the key with the current passphrase using the **crypto key decrypt rsa** command and encrypt the key once more to specify the new passphrase.

Encrypting and Locking Private Keys

To encrypt and lock the private keys, perform this task:

	Command	Purpose
Step 1	`ssl-proxy#` **`configure terminal`**	Enters global configuration mode.
Step 2	`ssl-proxy(config)#` **`crypto key encrypt`** `[`**`write`**`]` **`rsa`** `[`**`name`** *`key-name`*`]` **`passphrase`** *`passphrase`*	Encrypts the RSA keys. After this command is issued, the router can continue to use the key; the key remains unlocked. **Note** If the **write** keyword is not entered, the configuration must be manually written to NVRAM; otherwise, the encrypted key will be lost next time the router is reloaded.
Step 3	`ssl-proxy(config)#` **`exit`**	Exits global configuration mode.
Step 4	`ssl-proxy#` **`show crypto key mypubkey rsa`**	(Optional) Shows that the private key is encrypted (protected) and unlocked.
Step 5	`ssl-proxy#` **`crypto key lock rsa`** `[`**`name`** *`key-name`*`]` **`passphrase`** *`passphrase`*	(Optional) Locks the encrypted private key on a running router. **Note** After the key is locked, it cannot be used to authenticate the router to a peer device. This behavior disables any IPSec or SSL connections that use the locked key. Any existing IPSec tunnels created on the basis of the locked key will be closed. If all RSA keys are locked, SSH will automatically be disabled.

	Command	Purpose
Step 6	ssl-proxy# **show crypto key mypubkey rsa**	(Optional) Shows that the private key is protected and locked. The output will also show failed connection attempts via applications such as IKE, SSH, and SSL.
Step 7	ssl-proxy# **crypto key unlock rsa** [**name** *key-name*] **passphrase** *passphrase*	(Optional) Unlocks the private key.
Step 8	ssl-proxy# **configure terminal**	Enters global configuration mode
Step 9	ssl-proxy# **crypto key decrypt** [**write**] **rsa** [**name** *key-name*] **passphrase** *passphrase*	(Optional) Deletes the encrypted key and leaves only the unencrypted key. **Note** The **write** keyword immediately saves the unencrypted key to NVRAM. If the **write** keyword is not issued, the configuration must be manually written to NVRAM; otherwise, the key will remain encrypted the next time the router is reloaded.

The following example shows how to encrypt the RSA key "pki1-72a.cisco.com." Thereafter, the **show crypto key mypubkey rsa** command is issued to verify that the RSA key is encrypted (protected) and unlocked.

```
ssl-proxy(config)# crypto key encrypt rsa name pki1-72a.cisco.com passphrase cisco1234
ssl-proxy(config)# exit
ssl-proxy# show crypto key mypubkey rsa
% Key pair was generated at:00:15:32 GMT Jun 25 2003
Key name:pki1-72a.cisco.com
Usage:General Purpose Key
*** The key is protected and UNLOCKED. ***
Key is not exportable.
Key Data:
305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00E0CC9A 1D23B52C
CD00910C ABD392AE BA6D0E3F FC47A0EF 8AFEE340 0EC1E62B D40E7DCC
23C4D09E
03018B98 E0C07B42 3CFD1A32 2A3A13C0 1FF919C5 8DE9565F 1F020301 0001
% Key pair was generated at:00:15:33 GMT Jun 25 2003
Key name:pki1-72a.cisco.com.server
Usage:Encryption Key
Key is exportable.
Key Data:
307C300D 06092A86 4886F70D 01010105 00036B00 30680261 00D3491E 2A21D383
854D7DA8 58AFBDAC 4E11A7DD E6C40AC6 66473A9F 0C845120 7C0C6EC8 1FFF5757
3A41CE04 FDCB40A4 B9C68B4F BC7D624B 470339A3 DE739D3E F7DDB549 91CD4DA4
DF190D26 7033958C 8A61787B D40D28B8 29BCD0ED 4E6275C0 6D020301 0001
ssl-proxy#
```

The following example shows how to lock the key "pki1-72a.cisco.com." Thereafter, the **show crypto key mypubkey rsa** command is issued to verify that the key is protected (encrypted) and locked.

```
ssl-proxy# crypto key lock rsa name pki1-72a.cisco.com passphrase cisco1234
ssl-proxy# show crypto key mypubkey rsa
% Key pair was generated at:20:29:41 GMT Jun 20 2003
Key name:pki1-72a.cisco.com
Usage:General Purpose Key
*** The key is protected and LOCKED. ***
Key is exportable.
Key Data:
305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00D7808D C5FF14AC
```

```
0D2B55AC 5D199F2F 7CB4B355 C555E07B 6D0DECBE 4519B1F0 75B12D6F 902D6E9F
B6FDAD8D 654EF851 5701D5D7 EDA047ED 9A2A619D 5639DF18 EB020301 0001
ssl-proxy#
```

The following examples show how to verify that a key that is associated with a service is locked:

```
ssl-proxy# show ssl-proxy service
No context name provided, assuming context 'Default'...


Proxy Service Name  Context Name          Admin  Operation
                                          status status
service1            Default               up     down (No cert)

ssl-proxy# show ssl-proxy service service1
No context name provided, assuming context 'Default'...

Service id: 1, bound_service_id: 257
Virtual IP: 208.122.0.12, port: 443
Server IP: 209.0.207.203, port: 80
Virtual SSL Policy: test
Nat pool: np
rsa-general-purpose certificate trustpoint: test1
  Obsolete certificate chain for old connections:
    Certificate:
       Key Label: test1, 1024-bit, exportable
       Key Timestamp: 01:42:14 UTC Jul 12 2005
       Serial Number: 66A94DAC0000000000F7
    Root CA Certificate:
       Serial Number: 05A8229
  Keypair of trustpoint is locked
Context name: Default
Context Id  : 0
Admin Status: up
Operation Status: down
Proxy status: No certificate
```

Automatically Unlocking the Key Pair

In the event of a reload, you are able to unlock the key pair automatically.

You can store a passphrase in the startup configuration. The passphrase is encrypted using Advanced Encryption Standard (AES) encryption before it is saved to the configuration. When reading the passphrase from the configuration, the passphrase is treated as AES encrypted and is decrypted and used to unlock the key pair.

To unlock the keys automatically after a reload, perform this task in global configuration mode:

	Command	Purpose
Step 1	ssl-proxy(config)# **ssl-proxy crypto key unlock rsa** *key-name* **passphrase** *passphrase*	Unlocks the key automatically after a reload.
Step 2	ssl-proxy(config)# **exit**	Exits configuration mode.
Step 3	ssl-proxy# **copy system:running-config nvram:startup-config**	Saves the configuration.

Configuring Virtualization

When you configure virtualization, you are able to divide the SSL Services Module into multiple instances called contexts. Observe the following guidelines regarding virtualization:

- When you upgrade a previous SSL software release to SSL software release 3.x, your configuration is converted to a default context (Default) that is visible after the image is loaded on the SSL Services Module. After the conversion, you can configure additional contexts and assign them to different enterprises.
- You can configure multiple services per context.

Note You can configure up to 256 contexts; however, you can only have a maximum of 256 services configured per module, regardless of how many contexts you have configured.

- Services and policies are locally unique to a context.
- Statistics can be collected either per service or per context (by aggregating the statistics for all the services that are configured for that context).
- You can limit the maximum number of connections per context.

To configure the virtual context. perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)#` **`ssl-proxy context`** *`name`*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance. The context name is case-sensitive.
Step 2	`ssl-proxy(config-context)#` **`policy`** `{`**`health-probe tcp`** *`name`* `|` **`http-header`** `|` **`ssl`** *`policy-name`* `|` **`tcp`** *`policy-name`* `|` **`url-rewrite`**`}`	Configures the various policies. See the "Configuring Policies" section on page 4-1 for information on configuring policies.
Step 3	`ssl-proxy(config-context)#` **`maxconns`** *`connections`*	(Optional) Configures the maximum number of connections for this context. The default is 192000. Valid values are 1 to 192000.
Step 4	`ssl-proxy(config-context)#` **`natpool`** *`name start_ipaddr end_ipaddr`* **`netmask`** *`ipaddr`*	Configures the NAT pool settings.
Step 5	`ssl-proxy(config-context)#` **`pool ca`** *`name`*	Configures a pool of resources.
Step 6	`ssl-proxy(config-context)#` **`service`** *`service_name`*	Enters SSL proxy service subcommand mode and lets you configure the SSL client or server proxy service. See the "Configuring SSL Proxy Services" section on page 3-45 for information on SSL proxy services.
Step 7	`ssl-proxy(config-context)#` **`vrf-name`** *`name`*	Configures the VRF associated with this context.
Step 8	`ssl-proxy(config-context)#` **`description`** *`description`*	(Optional) Allows you to enter a short description for this context.
Step 9	`ssl-proxy(config-context)#` **`exit`**	Exit from SSL context mode.

Configuring SSL Proxy Services

You define SSL proxy services using the **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 3-45
- SSL Client Proxy Services, page 3-48
- IP Fragment Reassembly, page 3-50

SSL Server Proxy Services

To configure SSL server proxy services, perform this task:

	Command	Purpose
Step 1	ssl-proxy(config)# **ssl-proxy context** *name*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance. The context name is case-sensitive.
Step 2	ssl-proxy(config-context)# **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 3	ssl-proxy(config-ctx-ssl-proxy)# **virtual ipaddr** *ip_addr* [*mask_addr*][1] **protocol tcp port** *port* [**secondary**[2,3,4]]	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 4	ssl-proxy(config-ctx-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 5	ssl-proxy(config-ctx-ssl-proxy)# **virtual policy ssl** *ssl_policy_name*[5]	(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.
Step 6	ssl-proxy(config-ctx-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 4-5 for TCP policy parameters.
Step 7	ssl-proxy(config-ctx-ssl-proxy)# **server policy tcp** *tcp_policy_name*	(Optional) Applies a TCP policy to the server side of the proxy server. See the "Configuring TCP Policy" section on page 4-5 to configure the TCP policy.

	Command	Purpose
Step 8	ssl-proxy(config-ctx-ssl-proxy)# **policy http-header** *http_header_policy_name*	(Optional) Applies the HTTP header policy to proxy server. See the "HTTP Header Insertion" section on page 4-7 to configure the HTTP header insertion policy.
Step 9	ssl-proxy(config-ctx-ssl-proxy)# **policy url-rewrite** *url_rewrite_policy_name*	(Optional) Applies the URL rewrite policy. See the "Configuring URL Rewrite" section on page 4-11 to configure the URL rewrite policy.
Step 10	ssl-proxy(config-ctx-ssl-proxy)# **policy health-probe tcp** *name*	Applies the health probe policy. See the "Health Probe" section on page 4-13 to configure the health probe policy.
Step 11	ssl-proxy(config-ctx-ssl-proxy)# **trusted-ca** *ca_pool_name*	(Optional) Associates the trusted certificate authority pool with the proxy service. See the "Server Certificate Authentication" section on page 3-55 for information on certificate authority pools.
Step 12	ssl-proxy(config-ctx-ssl-proxy)# **authenticate verify** {**signature-only**[6] | **all**[7]}[**optional**]	(Optional) Enables server certificate authentication and specifies the form of verification. See the "Server Certificate Authentication" section on page 3-55 for information on server certificate authentication.
Step 13	ssl-proxy(config-ctx-ssl-proxy)# **nat** {**server** | **client** *natpool_name*}	(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 4-15.
Step 14	ssl-proxy(config-ctx-ssl-proxy)# **certificate rsa general-purpose trustpoint** *trustpoint_label*	Applies a trustpoint configuration to the proxy server[8]. **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 the "Declaring the Trustpoint" section on page 3-7 for information on configuring the trustpoint.
Step 15	ssl-proxy(config-ctx-ssl-proxy)# **inservice**	Sets the proxy server as administratively Up. **Note** 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.

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 Services 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.

[8] If the key (modulus) size is other than 512, 768, 1024, 1536, or 2048, you will receive an error and the trustpoint configuration is not applied. Replace the key by generating a key (using the same *key_label*) and specifying a supported modulus size, then repeat Step 14.

This example shows how to configure SSL proxy services:

```
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service proxy1
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ctx-ssl-proxy)# nat client t2
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint tp1
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-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 10.25.255.254:

```
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# 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-ctx-ssl-proxy)# server ipaddr 20.1.2.3 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
```

SSL Version 2.0 Forwarding

The SSL Services Module is not able to terminate SSL version 2.0 (SSLv2) connections. However, you can configure the SSL Services 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 Services 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:

`ssl-proxy(config-ctx-ssl-proxy)#` **`server ipaddr`** *`ip_addr`* **`protocol tcp port`** *`port`* **`sslv2`**[1]	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:

```
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service frontend
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 35.200.200.102 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 26.51.51.1 protocol tcp port 80
```

```
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 26.51.51.2 protocol tcp port 443 sslv2
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-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 client proxy services; SSL policies are configured at the **virtual** subcommand for SSL server proxy services.

To configure SSL client proxy services, perform this task:

	Command	Purpose
Step 1	ssl-proxy(config)# **ssl-proxy context** *name*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance.
Step 2	ssl-proxy(config-context)# **service** *proxy_name* **client**	Defines the name of the SSL proxy service. The **client** keyword configures the SSL client proxy service. **Note** You cannot use the same *service_name* for both the server proxy service and the client proxy service. **Note** The *proxy-name* value is case-sensitive.
Step 3	ssl-proxy(config-ctx-ssl-proxy)# **virtual ipaddr** *ip_addr* [*mask_addr*][1] **protocol tcp port** *port* [**secondary**[2,3,4]]	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 4	ssl-proxy(config-ctx-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 5	ssl-proxy(config-ctx-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 4-5 for TCP policy parameters.
Step 6	ssl-proxy(config-ctx-ssl-proxy)# **server policy ssl** *ssl_policy_name*[5]	(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.

	Command	Purpose

Step 7	`ssl-proxy(config-ctx-ssl-proxy)#`
	`server policy tcp` *`tcp_policy_name`*
		(Optional) Applies a TCP policy to the server side of the proxy server. See the "Configuring TCP Policy" section on page 4-5 for TCP policy parameters.

Step 8	`ssl-proxy(config-ctx-ssl-proxy)#`
	`policy http-header`
	`http_header_policy_name`
		(Optional) Applies the HTTP header policy to proxy server. See the "HTTP Header Insertion" section on page 4-7 to configure the HTTP header insertion policy.

Step 9	`ssl-proxy(config-ctx-ssl-proxy)#`
	`policy url-rewrite`
	`url_rewrite_policy_name`
		(Optional) Applies the URL rewrite policy. See the "Configuring URL Rewrite" section on page 4-11 to configure the URL rewrite policy.

Step 10	`ssl-proxy(config-ctx-ssl-proxy)#`
	`policy health-probe tcp` *`name`*
		Configures the health probe policy. See the "Health Probe" section on page 4-13 to configure the health probe policy.

Step 11	`ssl-proxy(config-ctx-ssl-proxy)#`
	`trusted-ca` *`ca_pool_name`*
		Associates the trusted certificate authority pool with the proxy service. See the "Client Certificate Authentication" section on page 3-51 for information on certificate authority pools.

Step 12	`ssl-proxy(config-ctx-ssl-proxy)#`
	`authenticate verify`
	`{`**`signature-only`**[6] `|` **`all`**[7]`}`
		Enables client certificate authentication and specifies the form of verification. See the "Client Certificate Authentication" section on page 3-51 for information on client certificate authentication.

Step 13	`ssl-proxy(config-ctx-ssl-proxy)#`
	`nat` `{`**`server`** `|` **`client`** *`natpool_name`*`}`
		(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.

Step 14	`ssl-proxy(config-ctx-ssl-proxy)#`
	`certificate rsa general-purpose`
	`trustpoint` *`trustpoint_label`*
		(Optional) Applies a trustpoint configuration to the client proxy.
		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 the "Declaring the Trustpoint" section on page 3-7 for information on configuring the trust point.

Step 15	`ssl-proxy(config-ctx-ssl-proxy)#`
	`inservice`
		Sets the proxy server as administratively Up.

[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 Services 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:

```
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service proxy1 client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# server policy ssl ssl1
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
```

IP Fragment Reassembly

The SSL Services Module receives IP fragments if an intermediate router (between the client and the SSL Services Module) cannot transmit the IP segment over its physical interface. In this case, the router fragments the packet. Each of these fragments might take a different path. The fragments are reasembled at the SSL Services Module.

The individual fragments might also be duplicated and take different paths which may further fragment the packet, which leads to packets that contain the same data bytes. Fragments may overlap if successive retransmissions use different packetizing but the same reassembly ID.

IP fragmentation is enabled by default. To adjust the IP fragment reassembly timer, perform this task:

Command	Purpose
`ssl-proxy(config)# ssl-proxy ip-frag-ttl [time]`	Configures the IP reassembly timeout. The default is 6 seconds. Valid values are from 3 to 120 seconds.
`ssl-proxy# show ssl-proxy ip-frag-ttl`	Displays the IP reassembly timeout.

Configuring Certificate Authentication

This section describes how to configure client and server authentication:

- Client Certificate Authentication, page 3-51
- Server Certificate Authentication, page 3-55
- Checking the Certificate Status, page 3-58
- Certificate Security Attribute-Based Access Control, page 3-65

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 Services Module does not accept the certificate.

When you enter the **verify signature-only** command, the SSL Services 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 Services Module sorts the certificate chain in order, ignoring any unrelated or redundant certificates. The SSL Services 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 Services Module cannot trust the top-most certificate, the chain is rejected.

If the SSL Services Module trusts the top-most certificate, then the SSL Services 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 Services Module selects one of these trustpoints. Depending on the certificate checking and ACL map configuration for this trustpoint, the SSL Services Module performs revocation and certificate attribute filtering. If the certificate checking denies the certificate, the SSL Services Module rejects the chain.
- If the certificate is a X509 version 3 certificate authority certificate, the SSL Services 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.

See the "Checking the Certificate Status" section on page 3-58 for information about on how to check certificate status using OCSP and CRLs.

Client Certificate Authentication

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

To authenticate the SSL client, the SSL Services 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 Services 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 Services Module trusts only the certificates issued by the certificate authorities that you configure in the certificate authority pool.

When you configure the SSL Services 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:

	Command	Purpose
Step 1	`ssl-proxy(config)#` **`ssl-proxy context`** *`name`*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)#` **`pool ca`** *`ca_pool_name`*	Creates the certificate authority pool. **Note** You can create up to eight pools.
Step 3	`ssl-proxy(config-ctx-ca-pool)#` **`ca trustpoint`** *`ca_trustpoint_label`*[1]	Adds a trusted certificate authority to the pool. **Note** You can add up to 16 trusted certificate authorities per pool.
Step 4	`ssl-proxy(config-ctx-ca-pool)#` **`exit`**	Returns to config mode.
Step 5	`ssl-proxy(config-context)#` **`service`** *`proxy_name`*	Defines the name of the SSL server proxy service.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)#` **`trusted-ca`** *`ca_pool_name`*	Associates the trusted certificate authority pool with the proxy service.

	Command	Purpose
Step 7	ssl-proxy(config-ctx-ssl-proxy)# **authenticate verify** {**signature-only**[2] | **all**[3]}	Enables client certificate authentication and specifies the form of verification. **Note** The **signature-only** keyword verifies the signature only, without looking up the CRL or matching ACL. The default is **all**.
Step 8	ssl-proxy(config-ctx-ssl-proxy)# **exit**	Exits from configuration mode.

[1] The SSL Services 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 Services 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 client certificate authentication verifying signature only:

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki 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-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----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 context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-auth-sig-only
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.1 protocol tcp port 443
```

```
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
```

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

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki 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-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----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 context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-auth-verify-all
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.2 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
```

Server Certificate Authentication

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

To authenticate the SSL server, the SSL Services 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 Services 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. A In this case, the SSL Services 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:

	Command	Purpose
Step 1	`ssl-proxy(config)#` **`ssl-proxy context`** *`name`*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)#` **`pool ca`** *`ca_pool_name`*	Creates the certificate authority pool. **Note** You can create up to eight pools.
Step 3	`ssl-proxy(config-ctx-ca-pool)#` **`ca trustpoint`** *`ca_trustpoint_label`*[1]	Adds a trusted certificate authority to the pool. **Note** You can add up to 16 trusted certificate authorities per pool.
Step 4	`ssl-proxy(config-ctx-ca-pool)#` **`exit`**	Returns to config mode.
Step 5	`ssl-proxy(config-context)#` **`service`** *`proxy_name`* **`client`**	Defines the name of the SSL server proxy service.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)#` **`trusted-ca`** *`ca_pool_name`*	Associates the trusted certificate authority pool with the proxy service. **Note** If you specify **signature-only** in Step 7, you must configure a certificate authority pool.

	Command	Purpose
Step 7	`ssl-proxy(config-ctx-ssl-proxy)#` **`authenticate verify`** {**`signature-only`**[2] | **`all`**[3]}	Enables client certificate authentication and specifies the form of verification. **Note** The **signature-only** keyword verifies the signature only, without looking up the CRL or matching ACL. The default is **all**.
Step 8	`ssl-proxy(config-ctx-ssl-proxy)#` **`exit`**	Exits from configuration mode.

1. The SSL Services 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 Services 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 server certificate authentication verifying signature only:

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki 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-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----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 context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-proxy-sig-only client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.3 protocol tcp port 81
```

```
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

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

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki 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-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----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 context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-proxy-verify-all client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.4 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

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

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
```

```
ssl-proxy(config)# crypto pki 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-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----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 context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-sending-client-cert client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.5 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.3 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

Checking the Certificate Status

There are two methods to check certificate status: Online Certificate Status Protocol (OCSP) and certificate revocation list (CRL).

The following sections give overview and configuration tasks for both methods.

- Certificate Revocation List, page 3-59
- Online Certificate Status Protocol, page 3-60

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.

If the certificate being validated contains a CRL distribution point (CDP) extension field, the module uses the CDP as the download path. The SSL Services 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 Services Module looks for a trustpoint that is associated with the certificate. If the module finds one or more associated trustpoints, the 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 key pair with the trustpoint. You can assign any existing key pair for this purpose. The key pair is used for signing the CRL download request.

The SSL Services 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).

Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) feature allows you to enable OCSP instead of certificate revocation lists (CRLs) to check certificate status. Unlike CRLs, which provide only periodic certificate status, OCSP can provide timely information regarding the status of a certificate.

OCSP allows a network administrator to configure a central OCSP server to collect and update CRLs from different certification authority (CA) servers; therefore, the devices within the network can rely on the OCSP server to check the certificate status without retrieving and caching each CRL for every device.

An OCSP server usually operates in either push or poll mode. You can configure a CA server to push revocation information to an OCSP server or configure an OCSP server to periodically download (poll) a CRL from the CA server. To ensure that timely certificate revocation status is obtained, you should carefully consider the “push and poll” interval.

When configuring an OCSP server to return the revocation status for a CA server, the OCSP server must be configured with an OCSP response signing certificate that is issued by that CA server. Ensure that the signing certificate is in the correct format, or the router will not accept the OCSP response. Refer to your OCSP manual for additional information.

The following is a sample OCSP response signing certificate (the extensions are in bold):

```
Certificate:
        Data:
            Version: v3
            Serial Number:0x14
            Signature Algorithm:MD5withRSA - 1.2.840.113549.1.1.4
            Issuer:CN=CA server,OU=PKI,O=Cisco Systems
            Validity:
                Not Before:Thursday, August 8, 2002 4:38:05 PM PST
                Not  After:Tuesday, August 7, 2003 4:38:05 PM PST
            Subject:CN=OCSP server,OU=PKI,O=Cisco Systems
            Subject Public Key Info:
                Algorithm:RSA - 1.2.840.113549.1.1.1
                Public Key:
                    Exponent:65537
                    Public Key Modulus:(1024 bits) :
                        <snip>

            Extensions:
                Identifier:Subject Key Identifier - 2.5.29.14
                    Critical:no
                    Key Identifier:
                        <snip>

                Identifier:Authority Key Identifier - 2.5.29.35
                    Critical:no
                    Key Identifier:
                        <snip>

                Identifier:OCSP NoCheck:- 1.3.6.1.5.5.7.48.1.5
                     Critical:no
                Identifier:Extended Key Usage:- 2.5.29.37
                     Critical:no
                     Extended Key Usage:
                     OCSPSigning
                Identifier:CRL Distribution Points - 2.5.29.31
                    Critical:no
                    Number of Points:1
                    Point 0
                        Distribution Point:
[URIName:ldap://CA-server/CN=CA server,OU=PKI,O=Cisco Systems]
        Signature:
            Algorithm:MD5withRSA - 1.2.840.113549.1.1.4
            Signature:
            <snip>
```

The following is a sample certificate that has OCSP URL defined AIA extension (the extensions are in bold):

```
Certificate:
        Data:
            Version:  v3
            Serial Number: 0xBB1
            Signature Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
            Issuer: CN=CA server,OU=PKI,O=Cisco Systems
            Validity:
                Not Before: Friday, February 21, 2003 6:41:07 PM PST America/Los_Angeles
                Not  After: Sunday, March 23, 2003 6:41:07 PM PST America/Los_Angeles
            Subject: CN=Client,OU=PKI,O=Cisco Systems
            Subject Public Key Info:
                Algorithm: RSA - 1.2.840.113549.1.1.1
                Public Key:
                    Exponent: 65537
                    Public Key Modulus: (512 bits) :
                    <snip>
            Extensions:
                Identifier: Key Usage: - 2.5.29.15
                    Critical: no
                    Key Usage:
                        Key Encipherment
                Identifier: Authority Key Identifier - 2.5.29.35
                    Critical: no
                    Key Identifier:
                    <snip>
                Identifier: CRL Distribution Points - 2.5.29.31
                    Critical: no
                    Number of Points: 1
                    Point 0
                        Distribution Point: [URIName: ldap://crl.cisco.com/CN=CA server,
                         OU=PKI,O=Cisco Systems]
                Identifier: Authority Info Access: - 1.3.6.1.5.5.7.1.1
                    Critical: no
                    Access Description:
                        Method #0: ocsp
                        Location #0: URIName: http://ocsp.cisco.com:81
        Signature:
            Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
            Signature:
            <snip>
```

Observe the following guidelines for OCSP:

- OCSP transports messages over HTTP, so there may be a time delay when you access the OCSP server. If the OCSP server is unavailable, certificate verification will fail.

- The increased certificate size may cause a problem for low-end routers when certificates are stored on NVRAM. Therefore, before you add the Authority Info Access (AIA) extension to a certificate, make sure that the increased size will not cause deployment problems.

Configuring the Revocation Check Method

To configure the SSL Services Module to check certificate status, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# crypto pki 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)# revocation-check method1 [method2[method3]]`	Checks the revocation status of a certificate. • **crl**—Certificate checking is performed by a CRL. This is the default option. • **none**—Certificate checking is ignored. • **ocsp**—Certificate checking is performed by an OCSP server. **Note** If a second and third method are specified, each method is used only if the previous method returns an error, such as a server being down. **Note** Since certificate checking is ignored when you enter the **none option**, you can only configure it at the end of the list.
Step 3	`ssl-proxy(ca-trustpoint)# ocsp url url`	(Optional) Specifies the URL of an OCSP server so that the trustpoint can check the certificate status. This URL overrides the URL of the OCSP server (if one exists) in Authority Info Access (AIA) extension of the certificate.

The following example shows how to configure the SSL Services Module to use the OCSP server at the HTTP URL “http://myocspserver:81.” If the server is down, revocation check is ignored.

```
ssl-proxy(config)# crypto pki trustpoint mytp
ssl-proxy(ca-trustpoint)# ocsp url http://myocspserver:81
ssl-proxy(ca-trustpoint)# revocation-check ocsp none
```

This following sections describe how to download and configure CRLs:

- Updating a CRL, page 3-63
- Entering X.500 CDP Information, page 3-63
- Entering a CRL Manually, page 3-64
- Displaying CRL Information, page 3-65
- Deleting a CRL, page 3-65

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 pki 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:

Command	Purpose
`ssl-proxy(config)#` **`crypto pki crl request`** *`trustpoint_label`*	Requests an updated CRL. This command replaces the currently stored CRL with the newest version of the CRL.

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 pki crl request tp-root
```

Entering X.500 CDP Information

You can enter the LDAP URL 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 certficate 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.

Note The CRL query must be sent on the administrative VLAN. To configure the SSL VLAN, refer to Adding the SSL Services Module to the Corresponding VLAN, page 2-8.

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

Command	Purpose
`ssl-proxy(ca-trustpoint)#` **`crl query`** *`urk`*	Allows you to specify the LDAP URL of the CA server that is used to query the CRL.

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:

	Command	Purpose
Step 1	`ssl-proxy(config)#` **`crypto pki certificate chain`** *`name`*	Enters certificate chain configuration mode; *name* specifies the name of the certificate authority.
Step 2	`ssl-proxy(config-cert-chain)#` **`crl`**	Allows you to enter the revocation list issued by the certificate authority manually.
Step 3	`ssl-proxy(config-pubkey)#` **`quit`**	Exits data entry mode.
Step 4	`ssl-proxy(config-cert-chain)#` **`end`**	Exits certificate chain configuration mode.

The following example shows how to manually enter a CRL:

```
ssl-proxy(config)# crypto pki certificate chain tp
ssl-proxy(config-cert-chain)# crl
Enter the CRL in hexidecimal representation....
ssl-proxy(config-pubkey)#     30 82 01 7E 30 81 E8 30 0D 06 09 2A 86 48 86 F7
ssl-proxy(config-pubkey)#     0D 01 01 05 05 00 30 16 31 14 30 12 06 03 55 04
ssl-proxy(config-pubkey)#     03 13 0B 69 6F 73 2D 72 6F 6F 74 20 43 41 17 0D
ssl-proxy(config-pubkey)#     30 33 31 32 31 31 32 33 33 37 35 34 5A 17 0D 30
ssl-proxy(config-pubkey)#     33 31 32 31 38 32 33 33 37 35 34 5A 30 81 A0 30
ssl-proxy(config-pubkey)#     12 02 01 04 17 0D 30 33 31 30 31 34 32 32 32 35
ssl-proxy(config-pubkey)#     30 34 5A 30 12 02 01 03 17 0D 30 33 31 30 31 34
ssl-proxy(config-pubkey)#     32 32 32 35 33 30 5A 30 12 02 01 05 17 0D 30 33
ssl-proxy(config-pubkey)#     31 31 31 33 31 39 30 39 33 36 5A 30 12 02 01 06
ssl-proxy(config-pubkey)#     17 0D 30 33 31 31 31 33 31 39 32 30 34 32 5A 30
ssl-proxy(config-pubkey)#     12 02 01 07 17 0D 30 33 31 31 31 33 32 32 30 35
ssl-proxy(config-pubkey)#     35 32 5A 30 12 02 01 08 17 0D 30 33 31 31 31 33
ssl-proxy(config-pubkey)#     32 32 34 34 32 30 5A 30 12 02 01 2B 17 0D 30 33
ssl-proxy(config-pubkey)#     31 31 31 33 32 33 33 36 35 37 5A 30 12 02 01 09
ssl-proxy(config-pubkey)#     17 0D 30 33 31 31 31 33 32 33 33 37 34 38 5A 30
ssl-proxy(config-pubkey)#     0D 06 09 2A 86 48 86 F7 0D 01 01 05 05 00 03 81
ssl-proxy(config-pubkey)#     81 00 67 DE 12 99 9F C5 DF 4A F8 24 76 CE 98 4F
ssl-proxy(config-pubkey)#     7C 5C 72 1C E0 00 A9 CE 08 6E 46 8F 4D 1B FA 8E
ssl-proxy(config-pubkey)#     C9 DE CF AC 13 7D 2F BF D4 A6 C2 7B E2 31 B1 EC
ssl-proxy(config-pubkey)#     99 83 54 B3 11 24 6F C3 C3 93 C4 53 38 B6 72 86
ssl-proxy(config-pubkey)#     0A 30 F2 95 71 AE 15 66 87 3E C1 7F 8B 46 6F A9
ssl-proxy(config-pubkey)#     77 D0 FF D4 FC 73 83 79 98 BD 40 DB C1 72 9D 95
ssl-proxy(config-pubkey)#     9B 57 D1 3C 2F EF B6 63 6B 5B E4 35 40 52 2D 3A
ssl-proxy(config-pubkey)#     19 1A 4E CA 70 C6 ED 49 7A 7C 01 88 B9 CA 14 7B
ssl-proxy(config-pubkey)#     0E 1F
ssl-proxy(config-pubkey)# quit
ssl-proxy(config-cert-chain)# end
```

Displaying CRL Information

To display information about the CRLs, enter this command:

Command	Purpose
`ssl-proxy# show crypto pki crls`	Displays the expiration date and time of each CRL in the database.

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

```
ssl-proxy# show crypto pki 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 pki certificate chain tp1
ssl-proxy(config-cert-chain)# no crl
ssl-proxy(config-cert-chain)# end
```

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