



Configuring SSL Services Secure Transactions

This chapter describes how to configure the CSM-S from the command line interface (CLI) of the module and contains these sections:

- [Configuring the Public Key Infrastructure, page 8-1](#)
- [Configuring the Certificate Authentication, page 8-40](#)

Configuring the Public Key Infrastructure

The SSL daughter card on the CSM-S uses the SSL protocol to enable secure transactions of data through privacy, authentication, and data integrity; the protocol relies upon certificates, public keys, and private keys.

The certificates, which are similar to digital ID cards, verify the identity of the server to the clients and the clients to the server. The certificates, which are issued by certificate authorities, 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.

The 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 daughter card 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 [“Configuring a Root CA \(Trusted Root\)” section on page 8-28](#) for more information.

The SSL daughter card authenticates certificates that it receives from external devices when you configure the SSL daughter card as an SSL server and you configure the server proxy to authenticate the client certificate, or when you configure the SSL daughter card as an SSL client. The SSL daughter card validates the start time, end time, and the signature on the certificate received.

A valid certificate may have been revoked if the key pair has been compromised. If a revocation check is necessary, the SSL daughter card downloads the certificate revocation list (CRL) from the certificate authority and looks up the serial number of the certificate received. See the [“Certificate Revocation List” section on page 8-47](#) for information on CRLs.

The certificate can also be filtered by matching certain certificate attribute values with access control list (ACL) maps. Only authenticated certificates that are issued by trusted certificate authorities are accepted. See the [“Certificate Security Attribute-Based Access Control”](#) section on page 8-51 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 daughter card.

However, the SSL daughter card 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 username and password, bank account number, credit card number, or mother’s maiden name.

If the certificate sender is an SSL client, the SSL daughter card 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 Insert”](#) section on page 10-14 for information on configuring HTTP header insertion. See the [“Client Certificate Authentication”](#) section on page 8-41 for information on configuring client certificate authentication.

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

- [Configuring the Keys and the Certificates, page 8-2](#)
- [Verifying the Certificates and the Trustpoints, page 8-27](#)
- [Configuring a Root CA \(Trusted Root\), page 8-28](#)
- [Backing Up the Keys and the Certificates, page 8-30](#)
- [Monitoring and Maintaining the Keys and Certificates, page 8-31](#)
- [Assigning a Certificate to a Proxy Service, page 8-32](#)
- [Renewing a Certificate, page 8-33](#)
- [Configuring the Automatic Certificate Renewal and Enrollment, page 8-36](#)
- [Enabling the Key and Certificate History, page 8-36](#)
- [Caching the Peer Certificates, page 8-37](#)
- [Configuring the Certificate Expiration Warning, page 8-38](#)

Configuring the Keys and the Certificates

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

- If you are using the 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 8-5 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 8-11 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 the Key Pairs and Certificates](#)” section on page 8-19 for details.

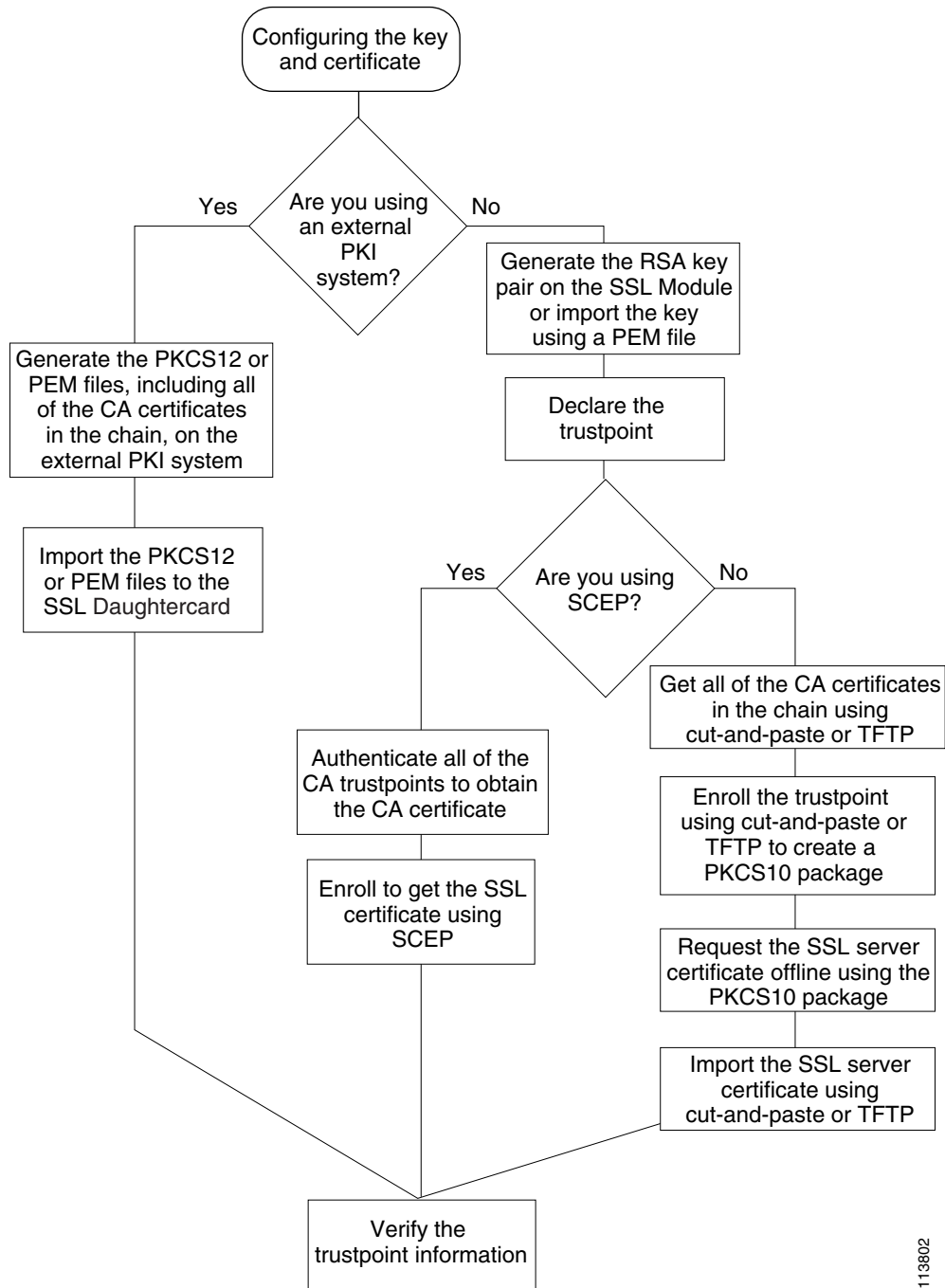
An external PKI system is a server or a PKI administration system that generates key pairs and enrolls for certificates from a certificate authority or a key and certificate archival system. The Public-Key Cryptography Standards (PKCS) specifies the transfer syntax for personal identity information, 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 8-1](#) for an overview on configuring keys and certificates.

Figure 8-1 Key and Certificate Configuration Overview



113802

Configuring the Trustpoint Using SCEP

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

- [Generating the RSA Key Pairs, page 8-5](#)
- [Declaring the Trustpoint, page 8-7](#)
- [Obtaining the Certificate Authority Certificate, page 8-8](#)
- [Requesting a Certificate, page 8-9](#)

Generating the 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 7-4](#).

RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Aldeman. The RSA algorithm is widely used by the certificate authorities and the 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 the 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 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 daughter card 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 8-9](#).

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 daughter card 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 and the key pairs to the private configuration file in the module NVRAM.

To generate the RSA key pairs, perform this task:

Command	Purpose
<pre>ssl-proxy(config)# crypto key generate rsa [usage-keysgeneral-keys label <i>key-label</i> [exportable¹] [modulus <i>size</i>]</pre>	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 the RSA keys, you are prompted to enter a modulus length in bits. The SSL daughter card 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 special-usage RSA keys:

```
crypto key generate rsa usage-keys
```

```
The name for the keys will be: myrouter.example.com
Choose the size of the key modulus in the range of 360 to 2048 for your Signature Keys.
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus[512]? <return>
```

```
Generating RSA keys.... [OK].
```

```
Choose the size of the key modulus in the range of 360 to 2048 for your Encryption Keys.
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus[512]? <return>
```

```
Generating RSA keys.... [OK].
```

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



Note

You cannot generate both special-usage and general-purpose keys; you can generate only one or the other.

```
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 [“Certificate Security Attribute-Based Access Control Examples”](#) section on page B-19.

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 ca trustpoint <i>trustpoint-label</i> ¹	Declares the trustpoint that your module should use. Enabling this command puts you in ca-trustpoint configuration mode.
Step 2	ssl-proxy(ca-trustpoint)# rsa keypair <i>key-label</i>	Specifies which key pair to associate with the certificate.
Step 3	ssl-proxy(ca-trustpoint)# enrollment [mode <i>ra</i>] [retry [period <i>minutes</i>] [count <i>count</i>]] url <i>url</i>	Specifies the enrollment parameters for your certificate authority.
Step 4	ssl-proxy(ca-trustpoint)# ip-address <i>server_ip_addr</i>	(Optional) Specifies the IP address of the proxy service that will use this certificate ² .
Step 5	ssl-proxy(ca-trustpoint)# crl [best-effort optional query <i>host:[port]</i>]	(Optional) Specifies how this trustpoint looks up a certificate revocation list when validating a certificate associated with this trustpoint. See the “Certificate Revocation List” section on page 8-47 for information on CRLs.
Step 6	ssl-proxy(ca-trustpoint)# subject-name <i>line</i> ^{3, 4}	(Optional) Configures the host name of the proxy service ⁵ .
Step 7	ssl-proxy(ca-trustpoint)# password <i>password</i>	(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 host name 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 ca 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)# crl optional
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# subject-name C=US; ST=California; L=San Jose; O=Cisco; OU=Lab;
CN=host1.cisco.com
ssl-proxy(ca-trustpoint)# end
ssl-proxy# ping example.cisco.com
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.0.0.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
ssl-proxy#
```

Obtaining the Certificate Authority Certificate

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



Note

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

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

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

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

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


Requesting a Certificate

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

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

Command	Purpose
<code>ssl-proxy(config)# crypto ca enroll trustpoint-label¹</code>	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 if your certificate needs to be revoked, so you must remember this password.



Note

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

This example shows how to request a certificate:

```
ssl-proxy(config)# crypto ca enroll PROXY1
%
% Start certificate enrollment..

% The subject name in the certificate will be: C=US; ST=California; L=San Jose; O=Cisco;
OU=Lab; CN=host1.cisco.com
% The subject name in the certificate will be: host.cisco.com
% The serial number in the certificate will be: 00000000
% The IP address in the certificate is 10.0.0.1

% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.
Fingerprint: 470DE382 65D8156B 0F84C2AF 4538B913

ssl-proxy(config)# end
```

After you configure the trustpoint, see the [“Verifying the Certificates and the Trustpoints”](#) section on [page 8-27](#) to verify the certificate and trustpoint information.

Example of Three-Tier Certificate Authority Enrollment

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

This 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 ca 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 ca 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 ca trustpoint tp-proxy1
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# password cisco
ssl-proxy(ca-trustpoint)# subject CN=ste.cisco.com
ssl-proxy(ca-trustpoint)# rsakeypair key1
ssl-proxy(ca-trustpoint)# show
  enrollment url tftp://10.1.1.3
  serial-number
  password 7 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 ca authenticate 3tier-root
Certificate has the following attributes:
Fingerprint:84E470A2 38176CB1 AA0476B9 C0B4F478
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate 3tier-sub1
Certificate has the following attributes:
Fingerprint:FE89FB0D BF8450D7 9934C926 6C66708D
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate tp-proxy1
Certificate has the following attributes:
Fingerprint:6E53911B E29AE44C ACE773E7 26A098C3
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.

```

Enrolling with the Third Level Certificate Authority

```

ssl-proxy(config)# crypto ca enroll tp-proxy1
%
% Start certificate enrollment ..

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

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

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

```

Manual Certificate Enrollment

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

- Your certificate authority does not support Simple Certificate Enrollment Protocol (SCEP)—This method is the most common method for sending and receiving requests and certificates.
- A network connection between the router and the certificate authority is not possible—This method is how a router running Cisco IOS software obtains its certificate.

Configure the Manual Certificate Enrollment (TFTP and cut-and-paste) feature as described at this URL: <http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t13/ftmancrt.htm>



Note

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

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

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

Configuring a Certificate Enrollment Using TFTP (One-Tier Certificate Authority)

To configure the certificate enrollment using TFTP, perform these steps:

Step 1 Configure the trustpoint:

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

Step 2 Request a certificate for the trustpoint:

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

% The fully-qualified domain name in the certificate will be: ssl-proxy.cisco.com
% The subject name in the certificate will be: ssl-proxy.cisco.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: 00000000
% Include an IP address in the subject name? [no]:
Send Certificate Request to tftp server? [yes/no]: yes
% Certificate request sent to TFTP Server
% The certificate request fingerprint will be displayed.
```

```
% The 'show crypto ca certificate' command will also show the fingerprint.
ssl-proxy(config)# Fingerprint: D012D925 96F4B5C9 661FEC1E 207786B7
!!
```

Step 3 Obtain the certificate that contains the public key of the certificate authority:

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

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

Step 4 Import the server certificate:

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

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

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

Configuring a Certificate Enrollment Using Cut-and-Paste (One-Tier Certificate Authority)

To configure the certificate enrollment using cut-and-paste, perform these steps:

Step 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]
```

Step 2 Configure the trustpoints:

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

a. Request a certificate for the trustpoint:

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

% The subject name in the certificate will be:CN=abc, OU=hss, O=cisco
```



```

MIIB7TCCAUVCAQQwDQYJKoZIhvcNAQEEBQAwUjELMAkGA1UEBhMCQVUxEzARBgNV
BAGTC1NvbWUtU3RhdGUxITAFBgNVBAoTGE1udGVybWV0IFdpZGdpdHMgUHR5IEExO
ZDELMAkGA1UEAxMCY2EwHhcNMjEwMDAxMzE2WhcNMjEwMDAxMzE2WjAs
MQQ4wDAYDVQKKEVjaXNjbzEMMAoGA1UECzMdAHNzMQwwCgYDVQQDEWhYmMwgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALt7O6tt301BVVK1qAE/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
```

Configuring a Certificate Enrollment Using TFTP (Three-Tier Certificate Authority)

To configure certificate enrollment using TFTP, perform these steps:

Step 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]
```

Step 2 Configure the trustpoint:

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

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

```
ssl-proxy(config)# crypto ca enroll test-3tier
%
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=test-3tier, OU=hss, O=Cisco
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be: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 ca certificate' command will also show the fingerprint.

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

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

Step 4 Use the CSR to acquire the SSL certificate offline from the third-level certificate authority.

Step 5 Authenticate the three certificate authorities (one root and two subordinate certificate authorities):

```
ssl-proxy(config)# crypto ca trustpoint test-1tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-1tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate test-1tier
Loading test-1tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1046 bytes]
```

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

```
ssl-proxy(config)# crypto ca trustpoint test-2tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-2tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate test-2tier
Loading test-2tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1554 bytes]
```

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

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

Step 6 Import the server certificate:

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

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

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

Configuring a Certificate Enrollment Using Cut-and-Paste (Three-Tier Certificate Authority)

To configure a certificate enrollment using cut-and-paste, perform these steps:

Step 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]
```

Step 2 Configure the trustpoint:

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

Step 3 Request a certificate for the trustpoint:

```
ssl-proxy(config)# crypto ca enroll tp-proxy1
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=test
% The fully-qualified domain name in the certificate will be:ssl-proxy.
% The subject name in the certificate will be:ssl-proxy.
% The serial number in the certificate will be:B0FFF14D
% Include an IP address in the subject name? [no]:no
Display Certificate Request to terminal? [yes/no]:yes
Certificate Request follows:

MIIBnDCCAQUCAQAwwZENMAsGA1UEAxMEdGVzdDEqMA8GA1UEBRMIQjBGRkYxNEQw
FwYJKoZIhvcNAQkCFgpzc2wtcHJveHkuMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQDFx1o19IXoAx4fyUhaXH6s4p5t9soIZ1gvLtVX6Fp6zfuX47os5TGJH/IX
zV9B4e5Kv+wLMD0AvTh+/tvyAP3TMpCdpHYosd2VaTIgExpHf4M5Ruh8IebVKV25
rraIpNiS0PvPLFcrw4UfJVNpsc2XBxBhpt+FS9y67LqlhfSN4wIDAQABoCEwHwYJ
KoZIhvcNAQkOMRIwEDAOBgNVHQ8BAf8EBAMCBaAwDQYJKoZIhvcNAQEEBQADgYEA
koIjd1KNJdKLMf33YELRd3MMw/ujJTuiT1J8RYVbwleE8JQf68TdTdKiYqzQcoMgsp
ez3vSPxXFZ/c6naXdVyrTikTX3GZ1mu+UOvV6/Jaf5QcXa9tAi3fgyguV7jQMPjk
Qj2GrwhXjcqZGOMBh6Kq6s5UPsIDgrL036t42B6B3EQ=

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

Redisplay enrollment request? [yes/no]:**no**

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

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

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



Note Use **tp-proxy1** to import the subordinate 2 certificate authority certificate.

```
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl op
```



```
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl op
ssl-proxy(ca-trustpoint)# exit
```

b. Import the root certificate authority certificate:

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

Enter the base 64 encoded CA certificate.

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

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcml5TERMA8GA1UEBxMlc2Fu
IGpvc2UxZjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLLEwNoc3MxIDAeBgNVBAMTF3Np
bXBz24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEwMTIxNDgwMl0xDTExMTEwMTIx
NTczOVowdTElMAkGA1UEBhMCMVVMxZARBgNVBAgTCmNhbGlmb3JuaWEExETAPBgNV
BACtCHNhbG1kb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdaHNzMSAwHgYD
VQDEExdzaW1wc29uLWRLdnRlc3Qtcml9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQWEibAnU1VqQNU0Wb94qnHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCml/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFYCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgnVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjb3NlMQ4wDAYDVQQKEwVjaXNjb3
cm9sbC9zaW1wc29uLWRLdnRlc3Qtcml9vdC1DQ55jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xZDZlJ0RW5yb2xsXHNpbXBz24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEeAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
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 the subordinate 1 certificate authority certificate:

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

Enter the base 64 encoded CA certificate.

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

```
-----BEGIN CERTIFICATE-----
MIIEtzCCA/mgAwIBAgIKGj0cBwAAAAADjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcml5TERMA8GA1UEBxMlc2FuIGpvc2Ux
ZjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLLEwNoc3MxIDAeBgNVBAMTF3NpbXBz24t
ZGV2dGVzdC1yb290LUNBMB4XDTAzMTEwMTIxNDgwMl0xDTExMTEwMTIxNTczOVow
dTElMAkGA1UEBhMCMVVMxZARBgNVBAgTCmNhbGlmb3JuaWEExETAPBgNVBACtCHNhb
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdaHNzMSAwHgYDVR0DEExdza
w1wc29uLWRLdnRlc3Qtc3ViMS1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQQDc
vV48nC2uukoSyGJ/GymCIEZXzMSzpbkYS7eWPaZYyijDhCikUUsMgFDRNFmQmUSA
rcWmPizFZc9PFumDa03vAgMBAAGjggJpMIICZTAQBgkrBgEeAYI3FQEEAwIBADAD
BgNVHQ4EFgQUWaaNN2U14BaBoU9mY+ncuHpP920wCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wga4GA1UdIwBpJCBo4AUJgYtQFMo130SBSiceehK9seDRrGh
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEWpYwXpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2l2Y28xDDAKBgNVBAAsTA2hzcEgMB4GA1UE
AxMxMjE1tCHNvbi1kZXZ0ZXN0LXJvb3Qtcml9vdC1DQ0GCEGnVmc1P4ve4Q5mUWCdWwX
AawgZcGAlUdHwSBjzCBjDBDoEGGp4Y9aHR0cDovL2Npc2NvLWw4ajZvaHBuc1xZDZlJ0RW5y
b2xsL3NpbXBz24tZGV2dGVzdC1yb290LUNBLmNybDBFoEOgQYY/ZmlsZTovL1xc
Y2l2Y28tbdhqnM9ocG5yXEN1cnRfbnJvbGxccc2l1tCHNvbi1kZXZ0ZXN0LXJvb3Qtc
ml9vdC1DQ0EuY3JsmIHIbGgrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Np
```

```
c2NvLWw4ajZvaHBuci9DZXJ0RW5yb2xsL2Npc2NvLWw4ajZvaHBuc19zaW1wc29u
LWRlbnRlc3Qtc9vdC1DQs5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2Nv
LWw4ajZvaHBuc1xZXXJ0RW5yb2xsXGNpc2NvLWw4ajZvaHBuc19zaW1wc29uLWRl
dnRlc3Qtc9vdC1DQs5jcnQwDQYJKoZIhvcNAQEFBQADQQA6kAV3Jx/BOR2h1Sp9
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 ca authenticate tp-proxy1
```

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

```
-----BEGIN CERTIFICATE-----
MIIESTCCA/OgAwIBAgIKHyiFxAaaaaaABjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvc25pYTERMA8GA1UEBxMlc2FuIGpvc2Ux
DjAMBGAoTBWNpc2NvMQwWcG9yYDZVQWwNoc3MxIDAeBgNVBAMTF3NpbXBzZ24t
ZGV2dGVzdC1zdWlxlWNhMB4XDTAzMTEwMTEwMTEwMTEwMTEwMTEwMTEwMTEw
dTELMaKGA1UEBhMCMVVMxZARBgNVBAGTCmNhbG1mb3JuaWEeXETAPEBGNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEWVjaXNjbzEMMAoGA1UECzMdAHNzMSAwHgYDVQDExdZ
aW1wc29uLWRlbnRlc3Qtc3ViMi1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQCQ7
ChZc0NYLBHf1sr/3Z4y6w5WoeioIpCOCSydhnd5wnwuethoyStVt9lR6i6lJwKl
d68Z8EoTg7ldaiV/WR/HAGMBAAGjggJjMIICXzAQBgkrBgEeAYI3FQEEAwIBADAD
BgNVHQ4EFgQU6FmJopqqzpbFMj6TaB2/wj1WlqEwCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wgagGA1UdIwSB0DCBnYAUWaaNN2U14BaBoU9mY+ncuHp922h
ear3MHUxCzAJBgNVBAYTA1VTMRMwEQYDVQQLIEwpcjYwXpZm9ybmlhMREwDwYDVQ
EhwhYw4gam9zZTEOMAwGA1UEChMFY2l2Y28xDDAKBgNVBAsTA2hzcEgMB4GA1UE
AxMxMzI1tcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCCo9HAcAAAAA4wgZcGA1UdHwSB
jzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xsL3Np
bXBzZ24tZGV2dGVzdC1zdWlxlWNhLmNybDBFoEOgQYY/ZmlsZTovL1xcY2l2Y28t
ZzI1dWE2bzRlXENlcnRfbnJvbGxccc2l2tcHNvbi1kZXZ0ZXN0LXN1YjEtY2EuY3Js
MIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwoZNaHR0cDovL2Npc2NvLWcy
NXVhNm80ZS9DZXJ0RW5yb2xsL2Npc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xsL29u
LWRlbnRlc3Qtc3ViMS1jYS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2Nv
LWcyNXVhNm80ZS9DZXJ0RW5yb2xsXGNpc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xs
L29uLWRlbnRlc3Qtc3ViMS1jYS5jcnQwDQYJKoZIhvcNAQEFBQADQQA6B8rvVCqVF
2cFw9/v51jGn7LQ6pUGT3bMRbOrgQKytTz/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 ca 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+gAwIBAgIKLmibDwAAAAACDANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvc25pYTERMA8GA1UEBxMlc2FuIGpvc2Ux
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvc25pYTERMA8GA1UEBxMlc2FuIGpvc2Ux
```

```
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIyLWNhMB4XDTAzMTEwOTIzNDUzNVoXDTA0MTEwMzIyMTQyMVow
PTErMA8GA1UEBRMIQjBGRkYxNEQxGTAXBgkqhkiG9w0BCQITCnNzbC1wcm94eS4x
DTALBgNVBAMTBHRlc3QwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMXHWiX0
hegDhh/JSPfcFqzInm32yghnWC8u1VfoWnrN+5fjuizlMYkf8hfNX0Hh7kq/7CUw
Af8EBAMCBaAwHQYDVR0OBBYEFcX1zcYHy01PNbfubnivi8d2VO22MIGoBgNVHSME
gaAwGZ2AF0hZiaKags6WxTI+k2gdv8I5VpahoXmkdzB1MQswCQYDVQQGEwJVUzET
MBEGA1UECBMKY2FsaWZvcn5pYTERMA8GA1UEBxMIc2FuIGpvc2UxDjAMBgNVBAoT
BWNpc2NvMQwwCgYDVQQLLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24tZGV2dGVzdC1z
dWIxLWNhggofKIXEAAAAAAGMIGXBGnVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9j
aXNjby1vam14Y25jenYvQ2VydEVucm9sbC9zaW1wc29uLWR1dnRlc3Qtc3ViMi1j
YS5jcmwwRaBDoEGGP2ZpbGU6Ly9cXGNpc2NvLW9qbXhjbmn6d1xdXJ0RW5yb2xs
XHNpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNybDcByAYIKwYBBQUHAQEgbswgbgw
WQYIKwYBBQUHMAKTWh0dHA6Ly9jaXNjby1vam14Y25jenYvQ2VydEVucm9sbC9j
aXNjby1vam14Y25jenZfc2l0cHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3J0MFsGCCsG
AQUFBzAChk9maWxloI8vXFxjaXNjby1vam14Y25jenZcQ2VydEVucm9sbFxaXNj
by1vam14Y25jenZfc2l0cHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3J0MA0GCSqGSIb3
DQEBBQUAA0EAtbxmUBOXz/hcrCc3hY7pa6q/LmLonXSL8cjAbv2I7A5QGyaNi5k9
8F1Ez1WoxW0J2C3/YsvIf4dYpsQWdKRJbQ==
-----END CERTIFICATE-----
```

```
% Router Certificate successfully imported
```

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

Importing and Exporting the 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.

These sections describe how to import or export key pairs and certificates:

- [Importing and Exporting a PKCS12 File, page 8-20](#)
- [Importing and Exporting the PEM Files, page 8-21](#)



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 [“Certificate Security Attribute-Based Access Control Examples” section on page B-19](#).



Note

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

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

the **crl optional** command for all the trustpoints associated with root CA and subordinate CA1.

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

Importing and Exporting a PKCS12 File

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



Note

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



Note

Imported key pairs cannot be exported.



Note

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

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

Command	Purpose
<pre>ssl-proxy(config)# crypto ca {import export} trustpoint_label pkcs12 {scp: ftp: nvram: rcp: tftp:} [pkcs12_filename¹] pass_phrase²</pre>	<p>Imports or exports a PKCS12 file.</p> <p>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.</p>

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

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

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

Password:password
Sending file modes:C0644 4379 TP2.p12
!
ssl-proxy(config)#
*Aug 22 12:30:00.531:%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 ca export TP1 pkcs12 scp: sky is blue
Address or name of remote host []? 10.1.1.1
Destination username [ssl-proxy]? admin-1
Destination filename [TP1]? TP1.p12
```

```
Password:

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

Password:
!
CRYPTO_PKI:Exported PKCS12 file successfully.
ssl-proxy(config)#
```

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

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

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

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

Writing /admin-1/pkcs12/PK-1024 !!
CRYPTO_PKI:Exported PKCS12 file successfully.
ssl-proxy(config)#
```

After you import the PKCS12 file, see the [“Verifying the Certificates and the Trustpoints”](#) section on [page 8-27](#) to verify the certificate and trustpoint information.

Importing and Exporting the PEM Files



Note

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



Note

Imported key pairs cannot be exported.



Note

If you are using SSH, we recommend using secure file transfer (SCP) 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
<pre>ssl-proxy(config)# crypto ca import trustpoint_label pem [exportable] {terminal url {scp: ftp: nvram: rcp: tftp:} usage-keys} pass_phrase^{1,2}</pre>	<p>Imports PEM files.</p> <p>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.</p>
<pre>ssl-proxy(config)# crypto ca export trustpoint_label pem {terminal url {scp: ftp: nvram: rcp: tftp:} [des 3des] pass_phrase^{1,2}</pre>	<p>Exports PEM files.</p> <p>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.</p>

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 the PEM files using TFTP:



Note

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

```
ssl-proxy(config)# crypto ca import TP5 pem url tftp://10.1.1.1/TP5 password
% Importing CA certificate...
Address or name of remote host [10.1.1.1]?
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 ca 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 the Certificates and the Trustpoints”](#) section on [page 8-27](#) to verify the certificate and trustpoint information.

Importing the 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.

To import the PEM files for three levels of certificate authority, perform these steps:

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

```
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca authenticate 3tier-root
```

Enter the base 64 encoded CA certificate.

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

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcn5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzbnQ2ZGV2dGVzdC1yb290LUNBMB4XDFAzMTEwMTIwNDgwMl0XDTEzMTExMTIx
NTczOVowdTElMAkGA1UEBhMCVVMxEzARBgNVBAGTCmNhbG1mb3JuaWEeETAPBgNV
BACgTCHNhbG1mb3N1MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECmDAHnZMSAwHgYD
VQDEExdzaW1wc29uLWRLdnRlc3Qtcml9vdc1DQTBcMA0GCSpGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnU1VqQNUn0Wb94qmHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAGHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBbYEFYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBod+GPWh0dHA6Ly9jaXNjb3N1S0Go2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRLdnRlc3Qtcml9vdc1DQ55jcmwwRaBDoEGGP2ZpbGU6Ly9c
```



```
VQQGEwJVUzETMBEGA1UECBMky2FsaWZvcml5pYTERMA8GA1UEBxMlc2FuIGpvc2Ux
DjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWlXlWNhMB4XDTAzmTEExMzIyMjI1Ml0XDTA0MTEExMzIyMTQyMVow
dTElMAkGA1UEBhMCMVVMxZARBgNVBAGTCmNhbG1mb3JuaWEExETAPBgNVBACThCNh
biBqb3N1MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdaHNzMSAwHgYDVQQDExdz
aW1wc29uLWR1dnRlc3Qtc3ViMS1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC7
ChZc0NYLBHf1sr/3Z4y6w5WoeioIpCOCsydhndb5wnwuehoyStvt9lr6i61jwK1
d68Z8EoTg71daiV/WR/HAGMBAAGjggJjMIICXzAQBgkrBgEAYI3FQEEAIBADAD
BgNVHQ4EFgQU6FmJopqzpbFMj6Tab2/wj1WlqEwCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wgagGA1UdIwSB0DCBnYAUWaaNN2U14BaBoU9mY+ncuHpP922h
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQLIEwepjYWxpZm9ybmlhMREwDwYDVQHQ
EwhzYW4gam9zZTEOMAwGA1UEChMfY21zY28xDDAKBgNVBAsTA2hzcEgMB4GA1UE
AxMxMz21tCHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCCh09HacAAAAAAA4wgZcGA1UdHwSE
jzCBjDBDoEGGP4Y9aHR0cDovL2Npc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xsL3Np
bXBzb24tZGV2dGVzdC1zdWlXlWNhLmNybDBFoEOgQYY/ZmlsZTovL1xcY21zY28t
ZzIldWE2bzRlXENlcnRFbnJvbGxccc21tCHNvbi1kZXZ0ZXN0LXN1YjEtY2EuY3Js
MIHIBGgrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwoAoZNaHR0cDovL2Npc2NvLWcy
NXVhNm80ZS9DZXJ0RW5yb2xsL2Npc2NvLWcyNXVhNm80ZV9zaW1wc29uLWR1dnRl
c3Qtc3ViMS1jYS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNCp2NvLWcyNXVh
Nm80ZV9zDZXJ0RW5yb2xsXGNCp2NvLWcyNXVhNm80ZV9zaW1wc29uLWR1dnRlc3Q
tc3ViMS1jYS5jcnQwQYJKoZIhvcNAQEFBQADQCCieB8rvVCqVF2cFw9/v51jGn7L
Q6pUGT3bMRbOrgQKytTz/Yx09156nYZHrvVuLzmz5CriI2saVx+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+YOipok40W0k5J/+dnRLwMjv+r10tr+LcCk
nBdR8zIOkOJObULLUOXFBM7oB3Dsk4Y3FBv8EAR3AdQizjevau4FiyQn+JfVZy+J
wctmvZnX0c0fevPsgID4dCPkeY6+I0DkxMyRiuy+nwIrwJlxVA2VIOrRJoBn1Ru
6/APef8JwpfnCgpcLYt/4Q+3Yj19EfrLjgiL6eSRki/6K51rV3eKbwOTyjjvXq5h
G0Q6dtNEoIvOg1Vad0CXeL+TxJySg4E630xIHkclDBsusGoUGLoz+OtaxApA2+5
WbKqR+ND1L1PmS8/ZL9LMPHUh9eOqZJjTe6NbxY7jeNHjAmpP7/WpB2f2kV/LZg
n2AV4GALBZtqXtreGayZzXpEA5J00lbzRZwf9JHA1diz/unW00/GH9LvCqA9015
YJGCrMI9US7Mwm8kIkiJqNgLtbPad5cOaieQe+Kncgcm18Hc7pFhDwXGG4RS40x
TSV/kIR4Gi7h8Lu71wZKtAWYHBPtUyTIpNsFUEdviTHXOSBw2LWNWzdYgppGoMT/t
ryuu01AC9YdBalAxY0DaqqpuXKzxfiw5QDbqZWVq3qAxXfLAtTgu/gFCuFQvbBG1
87H1c+nOQUq2nkpMphZLs13V0w/2yqg+q6rUydanFF+a5vRaLgX/Pgms92ZkZuDP
Z5qeKJmoURSlMYxDuhQD193RYxXJxOYIYrCrI/QaBpIH6QvUH60wWA==
```

```
-----END RSA PRIVATE KEY-----
```

```
quit
```

```
% Enter PEM-formatted certificate.
```

```
% End with a blank line or "quit" on a line by itself.
```

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

```
MIIEEXTCCBAegAwIBAgIKTJOcWgAAAAAACTANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMky2FsaWZvcml5pYTERMA8GA1UEBxMlc2FuIGpvc2Ux
DjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWlXlWNhMB4XDTAzmTEyNTIwMjI1Ml0XDTA0MTEExMzIyMTQyMVow
PjERMA8GA1UEBRMIQjBGRkYyMkUxKTAnBgkqhkiG9w0BCQITGnNpbXBzb24tNjUw
OS1zdGUuY21zY28uY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCKhRkm
38hSF710Wx1Ym8ixs2Hz/yjNw7tchtRPip0qCTJKW00gzZpp8dqaNi3s2GvVWb+t
Cgs010MZLlkyoj/9vT9MC7Zo3LOxYy9kD+6M9peUWMT4JLSD4Exxsds87JpPlbo0
o8WhYjvMor/bL30sW81y2RH2vppEMn9eLEN0vwIDAQABo4ICajCCAmYwCwYDVR0P
BAQDAgWgMB0GA1UdDgQWBBSx6uQ2sAR1cJzhBSiMu7xeu1n6AjCBqAYDVR0jBIGG
MIGdGBTOWYmimgrOlSuyPpNoHb/COVaWoaF5pHcWdTElMAkGA1UEBhMCMVVMxZAR
BgNVBAGTCmNhbG1mb3JuaWEExETAPBgNVBACThCNhbiBqb3N1MQ4wDAYDVQQKEwVj
aXNjbzEMMAoGA1UECzMdaHNzMSAwHgYDVQQDExdzZaW1wc29uLWR1dnRlc3Qtc3Vi
MS1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC7ChZc0NYLBHf1sr/3Z4y6w5Woe
ioIpCOCsydhndb5wnwuehoyStvt9lr6i61jwK1d68Z8EoTg71daiV/WR/HAGMBAAG
jggJjMIICXzAQBgkrBgEAYI3FQEEAIBADADBgNVHQ4EFgQU6FmJopqzpbFMj6Tab2/
wj1WlqEwCwYDVR0PBAQDAgHGMA8GA1UdEwEB/wQFMAMBAf8wgagGA1UdIwSB0DCB
nYAUWaaNN2U14BaBoU9mY+ncuHpP922heaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYD
VQQLIEwepjYWxpZm9ybmlhMREwDwYDVQHQEwhzYW4gam9zZTEOMAwGA1UEChMfY21
zY28xDDAKBgNVBAsTA2hzcEgMB4GA1UEAxMxMz21tCHNvbi1kZXZ0ZXN0LXJvb3Qt
Q0GCCh09HacAAAAAAA4wgZcGA1UdHwSEjzCBjDBDoEGGP4Y9aHR0cDovL2Npc2Nv
LWcyNXVhNm80ZS9DZXJ0RW5yb2xsL3NpbXBzb24tZGV2dGVzdC1zdWlXlWNhLmNyb
DBFoEOgQYY/ZmlsZTovL1xcY21zY28tZzIldWE2bzRlXENlcnRFbnJvbGxccc21tCH
Nvbi1kZXZ0ZXN0LXN1YjEtY2EuY3JsMEWg
```

```

Q6BBhj9maWx10i8vXFxjaXNjby1vam14Y25jenZcQ2VydEVucm9sbFxaW1wc29u
LWRldnRlc3Qtc3ViMi1jYS5jcmwwgcgGCCsGAQUFBwEBBIG7MIG4MFkGCCsGAQUF
BzAChk1odHRwOi8vY2l2Y28tb2pteGNuY3p2L0NlcnRFbnJvbGwvY2l2Y28tb2pt
eGNuY3p2X3NpbXBzb24tZGV2dGVzdC1zdWIyLWVhLmNydDBBbGgrBgEFBQcwAoZP
ZmlsZTovL1xcY2l2Y28tb2pteGNuY3p2XENlcnRFbnJvbGxcY2l2Y28tb2pteGNu
Y3p2X3NpbXBzb24tZGV2dGVzdC1zdWIyLWVhLmNydDANBgkqhkiG9w0BAQUFAANB
ABFh7XeLwvfBtjAR+e50aUH5KTGJDBeJppOmMFXnFakpgWop9Qg4cHRCQq7V0pAW
iA6VtJ0mpYgEIVNTAzAAHR4=
-----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#

```

Step 4 Display the certificate information (optional):

```

ssl-proxy# show crypto ca 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 ca 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 ca 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 the Certificates and the Trustpoints

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

	Command	Purpose
Step 1	ssl-proxy(ca-trustpoint)# show crypto ca certificates [<i>trustpoint_label</i>]	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 ca trustpoints [<i>trustpoint_label</i>]	Displays information about all trustpoints or the specified trustpoint.

Sharing the Keys and the Certificates

The SSL daughter card 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 daughter card does not impose any restrictions on sharing certificates among multiple proxy services and multiple SSL daughter cards. 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 your legal staff regarding business contractual aspects.

Some web browsers compare the subject name of the server certificate with the host name or the IP address that appears on the URL. If the subject name does not match the host name or IP address, a dialog box appears, prompting the user to verify and accept the certificate. To avoid this step, you should limit the sharing of certificates based on the host name or IP address.

Configuring a Root CA (Trusted Root)

To configure a trusted root, perform this task beginning in global configuration mode:

	Command	Purpose
Step 1	Router(config)# crypto ca trusted-root <i>name</i>	Configures a root with a selected name and enters trusted root configuration mode.
Step 2	Router(ca-root)# crl query <i>url</i>	(Optional) Queries the CRL published by the configured root with the LDAP ¹ URL.
Step 3	Router(ca-root)# exit	(Optional) Exits trusted root configuration mode.
Step 4	Router(config)# crypto ca trustpoint <i>name</i>	(Optional) Enters certificate authority identity configuration mode.
Step 5	Router(ca-identity) # crl optional	(Optional) Allows other peer certificates to be accepted by your switch even if the appropriate CRL is not accessible to your switch.
Step 6	Router(ca-identity)# exit	(Optional) Exits certificate authority identity configuration mode.
Step 7	Router(config)# crypto ca trusted-root <i>name</i>	(Optional) Enters trusted root configuration mode.
Step 8	Router(ca-root)# root { CEP <i>url</i> TFTP <i>server-hostname filename</i> }	Uses SCEP ² , with the given identity and URL, or TFTP to get a root certificate. Note Only use TFTP if your CA server does not support SCEP. Note When you are using TFTP, the server must be secured so that the downloading of the root certificate is not subject to any attack.
Step 9	Router(ca-root)# root proxy <i>url</i>	Defines the HTTP proxy server for getting a root certificate.

1. LDAP = Lightweight Directory Access Protocol.

2. SCEP = Simple Certificate Enrollment Protocol.

Saving Your Configuration



Caution


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



Tip

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

To save your configuration to NVRAM, perform this task:

Command	Purpose
<pre>ssl-proxy# copy [/erase] system:running-config nvram:startup-config</pre>	<p>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 does not need to query the certificate authority to obtain the certificates or to auto-enroll.</p> <p>Note For security reasons, you should enter the /erase keyword to erase the public and the private configuration files before updating the NVRAM. If you do not enter the /erase keyword, the key pairs from the old private configuration file may remain in the NVRAM.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> Caution When you enter the /erase keyword, 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.</p> </div>



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.

You should 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 hexadecimal 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 that are saved as hexadecimal dumps are also saved, resulting in two copies of the same certificate: one in hexadecimal 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 of proxy services (256) and certificates (356) configured, the output takes up to 7 minutes to display.

Erasing the Saved Configuration

To erase a saved configuration, perform one of these tasks:

	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 the Keys and the 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 the key pairs and the associated certificate chain by exporting each trustpoint to a PKCS12 file. You can then import the PKCS12 files to recover the keys and certificates.

Security Guidelines

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

- For each PKCS12, you must select a pass phrase that cannot be easily guessed and keep the pass phrase well protected. Do not store the PKCS12 file in clear form.
- The backup server must be secure. Allow only authorized personnel to access the backup server.

- When importing or exporting the PKCS12 file (in which you are required to enter a pass phrase), connect directly to the module console or use an SSH session.
- Use SCP for file transfers.

Monitoring and Maintaining the Keys and Certificates

These tasks are optional:

- [Deleting the RSA Keys from the Module, page 8-31](#)
- [Displaying the Keys and Certificates, page 8-32](#)
- [Deleting the Certificates from the Configuration, page 8-32](#)

Deleting the RSA Keys from the Module



Caution

Deleting the SSH key disables SSH on the module. If you delete the SSH key, generate a new key. For more information, see the [“Configuring SSH” section on page 7-4](#).

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.

To delete all RSA keys from the module, perform this task in global configuration mode:

Command	Purpose
<pre>ssl-proxy(config)# crypto key zeroize rsa [<i>key-label</i>]</pre>	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 ca enroll** command when you originally obtained the certificates.
- Manually remove the trustpoint from the configuration, as described in the [“Deleting the Certificates from the Configuration” section on page 8-32](#).

Displaying the Keys and Certificates

To display the keys and certificates, perform one of these tasks in EXEC mode:

Command	Purpose
<code>ssl-proxy# show crypto key mypubkey rsa</code>	Displays the RSA public keys for the module.
<code>ssl-proxy# show crypto ca certificates [trustpoint_label]</code>	Displays the information about the certificate, the certificate authority certificate, and any registration authority certificates.
<code>ssl-proxy# show running-config [brief]</code>	Displays the public keys and the certificate chains. If the brief keyword is specified, the hexadecimal dump of each certificate is not displayed.
<code>ssl-proxy# show ssl-proxy service proxy-name</code>	Displays the key pair and the serial number of the certificate chain used for a specified proxy service. Note The <i>proxy-name</i> value is case sensitive.

Deleting the Certificates from the Configuration

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

To delete the certificate from the module configuration, perform this task in global configuration mode:

Command	Purpose
<code>ssl-proxy(config)# no crypto ca trustpoint trustpoint-label</code>	Deletes the certificate.

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 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.

This example shows how to assign a trustpoint to a proxy service:

```
ssl-proxy# configure terminal
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# virtual ip 10.1.1.2 p tcp p 443
ssl-proxy(config-ssl-proxy)# server ip 20.0.0.3 p tcp p 80
```



```

ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# certificate rsa general trustpoint tp-1
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#
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#

```

This 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
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# certificate rsa general trustpoint tp-2
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#
ssl-proxy# show ssl-proxy service s2
Service id:6, bound_service_id:262
Virtual IP:10.1.1.2, port:443
Server IP:20.0.0.3, port:80
rsa-general-purpose certificate trustpoint:tp-2
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:k2
    Serial Number: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 that you 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 CSM-S.

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

This example shows that the 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 service s2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint t2
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#

ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
Certificate chain in use for new connections:
  Server Certificate:
    Key Label:k2
    Serial Number:1DFBB1FD000100000D48
  Root CA Certificate:
    Serial Number:313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Admin Status:up
Operation Status:up
```

This example shows that 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
```

This example shows that the existing and new connections use the old certificate until trustpoint t2 reenrolls. After trustpoint t2 reenrolls, the new connections use the new certificate; the 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 ca enroll t2
%
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=host1.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:00000000
% The IP address in the certificate is 10.1.1.1

% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

      Fingerprint: 6518C579 A0498063 C5795057 A6170 075

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
```

This example shows that 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
```

Configuring the Automatic Certificate Renewal and Enrollment

When you configure the 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 automatic 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	<code>ssl-proxy(config)# crypto ca trustpoint trustpoint-label</code>	Declares the trustpoint.
Step 2	<code>ssl-proxy(ca-trustpoint)# auto-enroll {renewal_percent regenerate}</code>	Enables automatic renewal and enrollment for the specified trustpoint. Note Valid values for <i>renewal_percent</i> 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	<code>ssl-proxy# show crypto ca timers</code>	Displays the time remaining before each timer expires.

This example shows how to enable automatic enrollment and auto renewal:

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

Enabling the 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	<code>ssl-proxy(config)# ssl-proxy pki history</code>	Enables key and certificate history.
Step 2	<code>ssl-proxy# show ssl-proxy certificate-history [service proxy_service]</code>	Displays key and certificate history records for all services or the specified service.

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 the Peer Certificates

You can configure the SSL daughter card to cache the authenticated server and client (peer) certificates. Caching the authenticated peer certificates saves time by not requiring the SSL daughter card to authenticate the same certificate again. The SSL daughter card 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 daughter card to incorrectly accept a certificate that should be denied. For instance, the SSL daughter card could cache a peer certificate and then within the specified timeout, download an updated CRL in which the certificate is listed.

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

For example, in a site-to-site VPN environment, where two SSL daughter cards 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.

The peer certificates that expire within the specified time interval are not cached. The SSL daughter card uses the 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 the different proxy services at a 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 situation is considered a cache miss, and the peer certificate is verified.

To cache the authenticated peer certificates, perform this task:

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

Configuring the Certificate Expiration Warning

You can configure the SSL daughter card 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 daughter card checks every 30 minutes for the expiration information for the following:

- All the proxy services
- The certificate authority certificates that are 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 daughter card 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 daughter card 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 that a proxy service certificate expiration warning is logged. For more information on configuring SNMP traps, see the [“Configuring SNMP Traps” section on page 7-24](#).

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

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

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:tpl-root.
```

Configuring the Certificate Authentication

This section describes how to configure client and server authentication:

- [Client Certificate Authentication, page 8-41](#)
- [Server Certificate Authentication, page 8-43](#)
- [Certificate Revocation List, page 8-47](#)
- [Certificate Security Attribute-Based Access Control, page 8-51](#)

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 daughter card does not accept the certificate.

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

If the SSL daughter card trusts the top-most certificate, then the SSL daughter card 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 daughter card selects one of these trustpoints. Depending on the CRL and ACL map configuration for this trustpoint, the SSL daughter card performs revocation and certificate attribute filtering. If the CRL or ACL checking denies the certificate, the SSL daughter card rejects the chain.
- If the certificate is a X509 version 3 certificate authority certificate, the SSL daughter card verifies that the Basic Constraints extension is present and valid. If the Basic Constraints extension is not present or valid, the chain is rejected.

If you verify only the signature, that verification process checks only the validity and signature of a minimum number of certificates in the chain. Verifying all performs more checking and validates all the certificates received, but takes longer and uses more CPU time.

You can download and update CRLs by entering CLI commands to reduce real-time delay. However, the CRL lookup is a slow process. See the [“Certificate Revocation List” section on page 8-47](#) for information about the CRLs.

Client Certificate Authentication

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

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



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 goes 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	<code>ssl-proxy(config)# ssl-proxy pool ca_pool_name</code>	Creates the certificate authority pool. Note You can create up to eight pools.
Step 2	<code>ssl-proxy(config-ssl-proxy)# ca trustpoint ca_trustpoint_label¹</code>	Adds a trusted certificate authority to the pool. Note You can add up to 16 trusted certificate authorities per pool.
Step 3	<code>ssl-proxy(config-ssl-proxy)# exit</code>	Returns to config mode.
Step 4	<code>ssl-proxy(config)# ssl-proxy service proxy_name</code>	Defines the name of the SSL server proxy service.
Step 5	<code>ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name</code>	Associates the trusted certificate authority pool with the proxy service.

	Command	Purpose
Step 6	<pre>ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only² all³}</pre>	<p>Enables the client certificate authentication and specifies the form of verification.</p> <p>Note The signature-only keyword verifies the signature only, without looking up the CRL or matching ACL. The default is all.</p>
Step 7	<pre>ssl-proxy(config-ssl-proxy)# exit</pre>	Exits from configuration mode.

1. The SSL daughter card 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, the 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 daughter card 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 a client certificate authentication verifying the signature only:

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

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

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcmlpYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBz24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEyNDgwMl0XDTEzMTExNDgwMl0x
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAGTCmNhbGlmb3JuaWEwETAPBgNV
BACgTCHNhbiBqb3N1MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRLdnRlc3Qtcml9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnU1VqQNU0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCml/raKJ/7ZAgMBAAGjgewwgekWCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVRO0BBYEFcyGLUBTKNd9EgUonHnoSvbg0axMIGX
BgNVHR8EgY8wgYwwQ6BB0D+GPWh0dHA6Ly9jaXNjb3N1MQ4wDAYDVQQDEwVjaXNj
cm9sbC9zaW1wc29uLWRLdnRlc3Qtcml9vdC1DQ5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xDZXJ0RW5yb2xsXHNpbXBz24tZGV2dGVzdC1yb290
LUNBMLNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqelwy
YjalelGZqLVu4bdVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
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(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-auth-sig-only
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.1 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.1 protocol tcp port 80
```

```

ssl-proxy(config-ssl-proxy) # certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy) # trusted-ca rootca
ssl-proxy(config-ssl-proxy) # authenticate verify signature-only
ssl-proxy(config-ssl-proxy) # inservice
ssl-proxy(config-ssl-proxy) # !

```

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

```

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

```

Enter the base 64 encoded CA certificate.

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

```

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADBl
MQswCQYDVOQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcn5pYTERMA8GA1UEBxMlcn2Fu
IGpvc2UxZDjAMBGNVBAoTBWVnc2NvMQwwCgYDVQQLLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEzMTExNDgwMl0XDTEzMTEzMTEx
NTczOVowdTElMAkGA1UEBhMCMVVMxEzARBgNVBAGTCmNhbG1mb3JuaWEwETAPBgNV
BACgTCHNhb3B3b3NlMQ4wDAYDVQQKEWVjaXNjbzEMMAoGA1UECmMDaHNzMSAwHgYD
VQDExdzaw1wc29uLWRLdnRlc3Qtc9vdc1DQTBcMA0GCsqGSIB3DQEBAQUAA0sA
MEgCQQCWEibAnU1VqQNU0Wb94qmHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekWcYDVR0PBAQDAGHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFYGLUBTKN9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBod+GPWh0dHA6Ly9jaXNjb3NlMQ4wDAYDVQQLLEwNoc3Mx
XGNpc2NvLWw4ajZvaHBuc1xDZXJORW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBMBNybDAQBgkrBgEeAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqelwy
YjalelGzqLVu4bdVMPo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
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(config) # ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool) # ca trustpoint rootca
ssl-proxy(config-ca-pool) # ^Z
ssl-proxy(config) # ssl-proxy service client-auth-verify-all
ssl-proxy(config-ssl-proxy) # virtual ipaddr 14.0.0.2 protocol tcp port 443
ssl-proxy(config-ssl-proxy) # server ipaddr 24.0.0.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy) # certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy) # trusted-ca rootca
ssl-proxy(config-ssl-proxy) # authenticate verify all
ssl-proxy(config-ssl-proxy) # inservice
ssl-proxy(config-ssl-proxy) # !

```

Server Certificate Authentication

When you configure the SSL daughter card as an SSL client (for example, for back-end encryption), the SSL daughter card always authenticates the SSL server.

To authenticate the SSL server, the SSL daughter card 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 daughter card accepts any certificate issued by any certificate authority trustpoint that has a certificate, is not listed in the CRL, and is not rejected by an ACL.

Optionally, you can create a trusted certificate authority pool and associate it with the proxy service. In this case, the SSL daughter card accepts only certificates that are 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	<code>ssl-proxy(config)# ssl-proxy pool ca_pool_name</code>	Creates the certificate authority pool. Note You can create up to eight pools.
Step 2	<code>ssl-proxy(config-ssl-proxy)# ca trustpoint ca_trustpoint_label¹</code>	Adds a trusted certificate authority to the pool. Note You can add up to 16 trusted certificate authorities per pool.
Step 3	<code>ssl-proxy(config-ssl-proxy)# exit</code>	Returns to config mode.
Step 4	<code>ssl-proxy(config)# ssl-proxy service proxy_name client</code>	Defines the name of the SSL client proxy service. Note Enter the client keyword to configure the SSL client proxy services.
Step 5	<code>ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name</code>	(Optional) Associates the certificate authority pool with the proxy service. Note If you specify signature-only in Step 6, you must configure a certificate authority pool.
Step 6	<code>ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only² all³}</code>	(Optional) Enables the server 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. You must configure a certificate authority pool to specify signature-only . The default is all .
Step 7	<code>ssl-proxy(config-ssl-proxy)# exit</code>	Exits configuration mode

1. The SSL daughter card 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, the 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 daughter card 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 the server certificate authentication verifying the signature only:

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

Enter the base 64 encoded CA certificate.

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

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcn5pYTERMA8GA1UEBxMlcn2Fu
IGpvc2UxDjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBz24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEzMTIxNDgwMl0XDTEzMTEzMTIx
NTczOVowdTElMAkGA1UEBhMCVVMxEzARBgNVBAGTCmNhbGlm3JuaWEExETAPBgNV
BACrTCHNhb3N1MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECjMDaHNzMSAwHgYD
VQDExdzawlwc29uLWRldnRlc3Qtcml9vdc1DQTBcMA0GCsqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnU1VqQNU0Wb94qmHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekWCwYDVR0PBAQDAGHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVROBBYEFcyGLUBTKND9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBod+GPWh0dHA6Ly9jaXNjb3N1MQ4wDAYDVQQKEwVjaXNjb3N1
cm9sbC9zaW1wc29uLWRldnRlc3Qtcml9vdc1DQs5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xDZXJ0RW5yb2xsXHNpbXBz24tZGV2dGVzdC1yb290
LUNBMLNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACqelwy
YjalelGzqLVu4bdVMPo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
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(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-proxy-sig-only client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.3 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !
```

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

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

Enter the base 64 encoded CA certificate.

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

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcn5pYTERMA8GA1UEBxMlcn2Fu
IGpvc2UxDjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBz24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTEzMTIxNDgwMl0XDTEzMTEzMTIx
NTczOVowdTElMAkGA1UEBhMCVVMxEzARBgNVBAGTCmNhbGlm3JuaWEExETAPBgNV
BACrTCHNhb3N1MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECjMDaHNzMSAwHgYD
VQDExdzawlwc29uLWRldnRlc3Qtcml9vdc1DQTBcMA0GCsqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnU1VqQNU0Wb94qmHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekWCwYDVR0PBAQDAGHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVROBBYEFcyGLUBTKND9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBod+GPWh0dHA6Ly9jaXNjb3N1MQ4wDAYDVQQKEwVjaXNjb3N1
cm9sbC9zaW1wc29uLWRldnRlc3Qtcml9vdc1DQs5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xDZXJ0RW5yb2xsXHNpbXBz24tZGV2dGVzdC1yb290
LUNBMLNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACqelwy
YjalelGzqLVu4bdVMPo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----
```

```

NTczOVowdTELMakGA1UEBhMCVVMxEzARBgNVBAGTCmNhbG1mb3JuaWEwETAPBgNV
BACtCHNhbiBqb3N1MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRLdnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBQUAA0sA
MEgCQQCWEibAnUlVqQNU0Wb94qnHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekWcYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFcYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwQ6BBOD+GPWh0dHA6Ly9jaXNjb3JuaWEwETAPBgNV
cm9sbC9zaW1wc29uLWRLdnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xZDZXXJ0RW5yb2xsXHNpbXBz24tZGVzdGVzdC1yb290
LUNBLmNybDAQBGRBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

```

```

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 ODC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

```

```

ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-proxy-verify-all client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.4 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# ^Z

```

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

```

ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca authenticate rootca

```

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

```

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFAADB1
MQswCQYDVQQGEwJVuzETMBEGA1UECBMKY2FsaWZvcn5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxZDjAMBGNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBz24tZGVzdGVzdC1yb290LUNBMB4XDTAzMTEExNDgwM1oXDTEzMTEExMTIx
NTczOVowdTELMakGA1UEBhMCVVMxEzARBgNVBAGTCmNhbG1mb3JuaWEwETAPBgNV
BACtCHNhbiBqb3N1MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECzMdaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRLdnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBQUAA0sA
MEgCQQCWEibAnUlVqQNU0Wb94qnHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP51
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekWcYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFcYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwQ6BBOD+GPWh0dHA6Ly9jaXNjb3JuaWEwETAPBgNV
cm9sbC9zaW1wc29uLWRLdnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuc1xZDZXXJ0RW5yb2xsXHNpbXBz24tZGVzdGVzdC1yb290
LUNBLmNybDAQBGRBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF41ktv9pCe6IO
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(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-proxy-sending-client-cert client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.5 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.3 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !

```

Certificate Revocation List

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



Note

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

This section describes how to download and configure CRLs:

- [Downloading the CRL, page 8-47](#)
- [Configuring the CRL Options, page 8-48](#)
- [Updating a CRL, page 8-49](#)
- [Entering the X.500 CDP Information, page 8-49](#)
- [Entering a CRL Manually, page 8-50](#)
- [Displaying the CRL Information, page 8-51](#)
- [Deleting a CRL, page 8-51](#)

Downloading the CRL

If the certificate being validated contains a CRL distribution point (CDP) extension field, the module uses the CDP as the download path. The SSL daughter card 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 daughter card looks for a trustpoint that is associated with the certificate. If the SSL daughter card finds one or more associated trustpoints, the SSL daughter card uses one of the trustpoints to determine the download path and protocol using the SCEP enrollment URL. If there is no SCEP enrollment URL, the validation fails.

**Note**

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

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

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

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

**Note**

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

Configuring the CRL Options

**Note**

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

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

You can configure two options for CRL lookup:

- Best-effort

If the SSL daughter card finds a CRL in the database and has not expired, then the SSL daughter card performs a CRL lookup. If the SSL daughter card does not find a CRL, the SSL daughter card attempts to download a CRL. However, if the CRL download fails, the SSL daughter card accepts the certificate.

- Optional

If the SSL daughter card finds a CRL in the database that has not expired, then the SSL daughter card performs a CRL lookup. If the SSL daughter card does not find the CRL, the SSL daughter card accepts the certificate. The SSL daughter card makes no attempt to download a CRL.

To configure the CRL options, perform this task:

	Command	Purpose
Step 1	<code>ssl-proxy(config)# crypto ca trustpoint trustpoint-label</code>	Declares the trustpoint that your module should use. Enabling this command puts you in ca-trustpoint configuration mode.
Step 2	<code>ssl-proxy(ca-trustpoint)# crl [best-effort optional]</code>	Specifies the CRL options.

Updating a CRL

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

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

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

To request immediate download of the latest CRL, perform this task:

Command	Purpose
<code>ssl-proxy(config)# crypto ca crl request trustpoint_label</code>	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 of the CRL.

This example shows how to download the CRL that is associated with the trustpoint “tp-root:”

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

Entering the X.500 CDP Information

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

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

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

then the two parts are combined to form the complete URL as follows:

```
ldap://10.1.1.1/CN=CRL,O=Cisco,C=US
```

**Note**

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 the CRL download cannot be performed.

To query the CRL with the X.500 URL, perform this task:

Command	Purpose
<code>ssl-proxy(ca-trustpoint)# crl query host:[port]</code>	Queries the CRL with the X.500 URL. The URL used to query the CRL must be an X.500 URL.

Entering a CRL Manually

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

To enter the CRL manually, perform this task:

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

This example shows how to enter a CRL manually:

```
ssl-proxy(config)# crypto ca certificate chain tp
ssl-proxy(config-cert-chain)# crl
Enter the CRL in hexadecimal 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 the CRL Information

To display information about the CRLs, perform this task:

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

```

Deleting a CRL



Note

The CRLs are deleted globally and not per trustpoint.

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

To delete a CRL, perform this task:

Command	Purpose
ssl-proxy(config-cert-chain)# no crl	Deletes the CRL.

This example shows how to delete the CRL:

```

ssl-proxy(config)# crypto ca 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 you to specify an access control list (ACL), so that you can create a certificate-based ACL.

For information on configuring the 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/ftcertacl.htm>

