CHAPTER 39

Configuring Certificates

This chapter describes how to configure certificates. CAs are responsible for managing certificate requests and issuing digital certificates. A digital certificate contains information that identifies a user or device. Some of this information can include a name, serial number, company, department, or IP address. A digital certificate also contains a copy of the public key for the user or device. A CA can be a trusted third party, such as VeriSign, or a private (in-house) CA that you establish within your organization.

This chapter includes the following sections:

Public Key Cryptography

This section includes the following topics:

About Public Key Cryptography

Digital signatures, enabled by public key cryptography, provide a means to authenticate devices and users. In public key cryptography, such as the RSA encryption system, each user has a key pair containing both a public and a private key. The keys act as complements, and anything encrypted with one of the keys can be decrypted with the other.

In simple terms, a signature is formed when data is encrypted with a private key. The signature is attached to the data and sent to the receiver. The receiver applies the public key of the sender to the data. If the signature sent with the data matches the result of applying the public key to the data, the validity of the message is established.

This process relies on the receiver having a copy of the public key of the sender and having a high degree of certainty that this key belongs to the sender, not to someone pretending to be the sender.

Obtaining the public key of a sender is normally handled out-of-band or through an operation done at installation. For instance, most web browsers are configured with the root certificates of several CAs by default. For VPN, the IKE protocol, a component of IPSec, can use digital signatures to authenticate peer devices before setting up security associations.

Certificate Scalability

Without digital certificates, you must manually configure each IPSec peer for every peer with which it communicates, and every new peer you add to a network would thus require a configuration change on every peer with which you need it to communicate securely.

When you use digital certificates, each peer is enrolled with a CA. When two peers attempt to communicate, they exchange certificates and digitally sign data to authenticate each other. When a new peer is added to the network, you enroll that peer with a CA and none of the other peers need modification. When the new peer attempts an IPSec connection, certificates are automatically exchanged and the peer can be authenticated.

With a CA, a peer authenticates itself to the remote peer by sending a certificate to the remote peer and performing some public key cryptography. Each peer sends its unique certificate which was issued by the CA. This process works because each certificate encapsulates the public key for the associated peer and each certificate is authenticated by the CA, and all participating peers recognize the CA as an authenticating authority. This is called IKE with an RSA signature.

The peer can continue sending its certificate for multiple IPSec sessions, and to multiple IPSec peers, until the certificate expires. When its certificate expires, the peer administrator must obtain a new one from the CA.

CAs can also revoke certificates for peers that no longer participate in IPSec. Revoked certificates are not recognized as valid by other peers. Revoked certificates are listed in a CRL, which each peer may check before accepting a certificate from another peer.

Some CAs have an RA as part of their implementation. An RA is a server that acts as a proxy for the CA so that CA functions can continue when the CA is unavailable.

About Key Pairs

Key pairs are RSA keys.

- RSA keys can be used for SSH or SSL.
- SCEP enrollment supports the certification of RSA keys.
- For the purposes of generating keys, the maximum key modulus for RSA keys is 2048. The default size is 1024.
- For signature operations, the supported maximum key size is 4096 bits.
- You can generate a general purpose RSA key pair, used for both signing and encryption, or you can generate separate RSA key pairs for each purpose.

  Separate signing and encryption keys helps reduce exposure of the keys. This is because SSL uses a key for encryption but not signing but IKE uses a key for signing but not encryption. By using separate keys for each, exposure of the keys is minimized.

About Trustpoints

Trustpoints let you manage and track CAs and certificates. A trustpoint is a representation of a CA or identity pair. A trustpoint contains the identity of the CA, CA-specific configuration parameters, and an association with one enrolled identity certificate.

After you have defined a trustpoint, you can reference it by name in commands requiring that you specify a CA. You can configure many trustpoints.

Note If a security appliance has multiple trustpoints that share the same CA, only one of these trustpoints sharing the CA can be used to validate user certificates. Use the **support-user-cert-validation** command to control which trustpoint sharing a CA is used for validation of user certificates issued by that CA.

For automatic enrollment, a trustpoint must be configured with an enrollment URL and the CA that the trustpoint represents must be available on the network and must support SCEP.

You can export and import the keypair and issued certificates associated with a trustpoint in PKCS12 format. This is useful if you wish to manually duplicate a trustpoint configuration on a different security appliance.

About Revocation Checking

When a certificate is issued, it is valid for a fixed period of time. Sometimes a CA revokes a certificate before this time period expires; for example, due to security concerns or a change of name or association. CAs periodically issue a signed list of revoked certificates. Enabling revocation checking forces the security appliance to check that the CA has not revoked a certificate every time it uses that certificate for authentication.

Note Make sure that you limit the validity period of the certificate to less than the recommended end date of 03:14:08 UTC, January 19, 2038.

When you enable revocation checking, during the PKI certificate validation process the security appliance checks certificate revocation status. It can use either CRL checking or Online Certificate Status Protocol or both, with the second method you set in effect only when the first method returns an error, for example, that the server is unavailable.

With CRL checking, the security appliance retrieves, parses, and caches Certificate Revocation Lists, which provide a complete list of revoked certificates. OCSP offers a more scalable method of checking revocation status in that it localizes certificate status on a Validation Authority, which it queries for the status of a specific certificate.

About CRLs

Certificate Revocation Lists provide the security appliance with one means of determining whether a certificate that is within its valid time range has been revoked by its issuing CA. CRL configuration is a part of the configuration of a trustpoint.

You can configure the security appliance to make CRL checks mandatory when authenticating a certificate (**revocation-check crl** command). You can also make the CRL check optional by adding the **none** argument (**revocation-check crl none** command), which allows the certificate authentication to succeed when the CA is unavailable to provide updated CRL data.

The security appliance can retrieve CRLs from CAs using HTTP, SCEP, or LDAP. CRLs retrieved for each trustpoint are cached for a length of time configurable for each trustpoint.

When the security appliance has cached a CRL for more than the length of time it is configured to cache CRLs, the security appliance considers the CRL too old to be reliable, or "stale". The security appliance attempts to retrieve a newer version of the CRL the next time a certificate authentication requires checking the stale CRL.

The security appliance caches CRLs for a length of time determined by the following two factors:

- The number of minutes specified with the **cache-time** command. The default value is 60 minutes.
- The NextUpdate field in the CRLs retrieved, which may be absent from CRLs. You control whether the security appliance requires and uses the NextUpdate field with the **enforcenextupdate** command.

The security appliance uses these two factors as follows:

- If the NextUpdate field is not required, the security appliance marks CRLs as stale after the length of time defined by the **cache-time** command.
- If the NextUpdate field is required, the security appliance marks CRLs as stale at the sooner of the two times specified by the **cache-time** command and the NextUpdate field. For example, if the cache-time command is set to 100 minutes and the NextUpdate field specifies that the next update is 70 minutes away, the security appliance marks CRLs as stale in 70 minutes.

If the security appliance has insufficient memory to store all CRLs cached for a given trustpoint, it deletes the least recently used CRL to make room for a newly retrieved CRL.

For information about configuring CRL behavior for a trustpoint, see the "Configuring CRLs for a Trustpoint" section on page 39-13.

About OCSP

Online Certificate Status Protocol provides the security appliance with a means of determining whether a certificate that is within its valid time range has been revoked by its issuing CA. OCSP configuration is a part of the configuration of a trustpoint.

OCSP localizes certificate status on a Validation Authority (an OCSP server, also called the *responder*) which the security appliance queries for the status of a specific certificate. It provides better scalability and more up-to-date revocation status than does CRL checking. It helps organizations with large PKI installations deploy and expand secure networks.

You can configure the security appliance to make OCSP checks mandatory when authenticating a certificate (**revocation-check ocsp** command). You can also make the OCSP check optional by adding the **none** argument (**revocation-check ocsp none** command), which allows the certificate authentication to succeed when the Validation Authority is unavailable to provide updated OCSP data.

Our implementation of OCSp provides three ways to define the OCSP server URL. The security appliance uses these servers in the following order:

1. The OCSP URL defined in a match certificate override rule (**match certificate** command).
2. The OCSP URL configured in the **ocsp url** command.
3. The AIA field of the client certificate.

Note To configure a trustpoint to validate a self-signed OCSP responder certificate, you import the self-signed responder certificate into its own trustpoint as a trusted CA certificate. Then you configure the **match certificate** command in the client certificate validating trustpoint to use the trustpoint that contains the

self-signed OCSP responder certificate to validate the responder certificate. The same applies for configuring validating responder certificates external to the validation path of the client certificate.

The OCSP server (responder) certificate typically signs the OCSP response. After receiving the response, the security appliance tries to verify the responder certificate. The CA normally sets the lifetime of its OCSP responder certificate to a relatively short period to minimize the chance of it being compromised.The CA typically also includes an ocsp-no-check extension in the responder certificate indicating that this certificate does not need revocation status checking. But if this extension is not present, the security appliance tries to check its revocation status using the same method specified in the trustpoint. If the responder certificate is not verifiable, revocation checks fails. To avoid this possibility, configure **revocation-check none** in the responder certificate validating trustpoint, while configuring r**evocation-check ocsp** for the client certificate.

Supported CA Servers

The security appliance supports the following CA servers:

- Cisco IOS CS
- Baltimore Technologies
- Entrust
- Microsoft Certificate Services
- Netscape CMS
- RSA Keon
- VeriSign

Certificate Configuration

This section describes how to configure the security appliance with certificates and other procedures related to certificate use and management.

This section includes the following topics:

Preparing for Certificates

Before you configure a security appliance with certificates, ensure that the security appliance is configured properly to support certificates. An improperly configured security appliance can cause enrollment to fail or for enrollment to request a certificate containing inaccurate information.

To prepare a security appliance for certificates, perform the following steps:

Step 1 Ensure that the hostname and domain name of the security appliance are configured correctly. You can use the **show running-config** command to view the hostname and domain name as currently configured.

For information about configuring the hostname, see the "Setting the Hostname" section on page 8-2.

For information about configuring the domain name, see the "Setting the Domain Name" section on page 8-2.

Step 2 Be sure that the security appliance clock is set accurately before configuring the CA. Certificates have a date and time that they become valid and that they expire. When the security appliance enrolls with a CA and gets a certificate, the security appliance checks that the current time is within the valid range for the certificate. If it is outside that range, enrollment fails.

For information about setting the clock, see the "Setting the Date and Time" section on page 8-2.

Configuring Key Pairs

This section includes the following topics:

Generating Key Pairs

Key pairs are RSA keys, as discussed in the "About Key Pairs" section on page 39-2. You must generate key pairs for the types of certification you want to use.

To generate key pairs, perform the following steps:

Step 1 Generate the types of key pairs needed for your PKI implementation. To do so, perform the following steps, as applicable:

a. If you want to generate RSA key pairs, use the **crypto key generate rsa** command.

```
hostname/contexta(config)# crypto key generate rsa
```

If you do not use additional keywords this command generates one general purpose RSA key pair. Because the key modulus is not specified, the default key modulus of 1024 is used. You can specify other modulus sizes with the **modulus** keyword. You can also assign a label to each key pair using the **label** keyword. The label is referenced by the trustpoint that uses the key pair. If you do not assign a label, the key pair is automatically labeled <Default-RSA-Key>.

```
hostname/contexta(config)# crypto key generate rsa label key-pair-label
```

Step 2 (Optional) Use the **show crypto key mypubkey** command to view key pair(s). The following example shows an RSA general-purpose key:

```
hostname/contexta(config)# show crypto key mypubkey
Key pair was generated at: 16:39:47 central Feb 10 2005
Key name: <Default-RSA-Key>
 Usage: General Purpose Key
 Modulus Size (bits): 1024
 Key Data:
```

```
  30819f30 0d06092a 864886f7 0d010101 05000381 8d003081 89028181 00ea51b7
  0781848f 78bccac2 4a1b5b8d 2f3e30b4 4cae9f86 f4485207 159108c9 f5e49103
  9eeb0f5d 45fd1811 3b4aafce 292b3b64 b4124a6f 7a777b08 75b88df1 8092a9f8
  5508e9e5 2c271245 7fd1c0c3 3aaf1e04 c7c4efa4 600f4c4a 6afe56ad c1d2c01c
  e08407dd 45d9e36e 8cc0bfef 14f9e6ac eca141e4 276d7358 f7f50d13 79020301 0001
Key pair was generated at: 16:34:54 central Feb 10 2005
```

Step 3 Save the key pair you have generated. To do so, save the running configuration by entering the **write memory** command.

Removing Key Pairs

To remove key pairs, use the **crypto key zeroize command in global configuration mode.**

The following example removes RSA key pairs:

```
hostname(config)# crypto key zeroize rsa
WARNING: All RSA keys will be removed.
WARNING: All device certs issued using these keys will also be removed.

Do you really want to remove these keys? [yes/no] y
hostname(config)#
```

Configuring Trustpoints

For information about trustpoints, see the "About Trustpoints" section on page 39-3.

To configure a trustpoint, perform the following steps:

Step 1 Create a trustpoint corresponding to the CA from which the security appliance needs to receive its certificate.

```
hostname/contexta(config)# crypto ca trustpoint trustpoint
```

For example, to declare a trustpoint called Main:

```
hostname/contexta(config)# crypto ca trustpoint Main
hostname/contexta(config-ca-trustpoint)#
```

Upon entering this command, you enter the Crypto ca trustpoint configuration mode.

Step 2 Specify the enrollment method to be used with this trustpoint.

To specify the enrollment method, do one of the following items:

- To specify SCEP enrollment, use the **enrollment url** command to configure the URL to be used for SCEP enrollment with the trustpoint you declared. For example, if the security appliance requests certificates from trustpoint Main using the URL http://10.29.67.142:80/certsrv/mscep/mscep.dll, then the command would be as follows:

  ```
  hostname/contexta(config-ca-trustpoint)# enrollment url
  http://10.29.67.142:80/certsrv/mscep/mscep.dll
  ```

- To specify manual enrollment, use the **enrollment terminal** command to indicate that you will paste the certificate received from the CA into the terminal.

Step 3 As needed, specify other characteristics for the trustpoint. The characteristics you need to define depend upon your CA and its configuration. You can specify characteristics for the trustpoint using the following commands. Refer to the *Cisco Security Appliance Command Reference* for complete descriptions and usage guidelines of these commands.

- **accept-subordinates**—Indicates whether CA certificates subordinate to the CA associated with the trustpoint are accepted if delivered during phase one IKE exchange when not previously installed on the device.
- **crl required | optional | nocheck**—Specifies CRL configuration options. When you enter the **crl** command with the **optional** keyword included within the command statement, certificates from peers can still be accepted by your security appliance even if the CRL is not accessible to your security appliance.

  Note If you chose to enable required or optional CRL checking, be sure you configure the trustpoint for CRL managemen2t, which should be completed after you have obtained certificates. For details about configuring CRL management for a trustpoint, see the "Configuring CRLs for a Trustpoint" section on page 39-13.

- **crl configure**—Enters CRL configuration mode.
- **default enrollment**—Returns all enrollment parameters to their system default values. Invocations of this command do not become part of the active configuration.
- **email** *address*—During enrollment, asks the CA to include the specified email address in the Subject Alternative Name extension of the certificate.
- **enrollment retry period** —(Optional) Specifies a retry period in minutes. This characteristic only applies if you are using SCEP enrollment.
- **enrollment retry count**—(Optional) Specifies a maximum number of permitted retries. This characteristic only applies if you are using SCEP enrollment.
- **enrollment terminal**—Specifies cut and paste enrollment with this trustpoint.
- **enrollment url** *URL*—Specifies automatic enrollment (SCEP) to enroll with this trustpoint and configures the enrollment URL.
- **fqdn** *fqdn*—During enrollment, asks the CA to include the specified fully qualified domain name in the Subject Alternative Name extension of the certificate.
- **id-cert-issuer**—Indicates whether the system accepts peer certificates issued by the CA associated with this trustpoint.
- **ip-address** *ip-address*—During enrollment, asks the CA to include the IP address of the security appliance in the certificate.
- **keypair** *name*—Specifies the key pair whose public key is to be certified.
- **match certificate map**—Configures OCSP URL overrides and trustpoints to use to validate OCSP responder certificates
- **ocsp disable-nonce**—Disable the nonce extension on an OCSP request; the nonce extension cryptographically binds requests with responses to avoid replay attacks.
- **ocsp url**—Configures an OCSP server for the security appliance to use to check all certificates associated with a trustpoint rather than the server specified in the AIA extension of the client certificate.
- **password** *string*—Specifies a challenge phrase that is registered with the CA during enrollment. The CA typically uses this phrase to authenticate a subsequent revocation request.

- **revocation-check**—Sets one or more methods for revocation checking: CRL, OCSP, and none.
- **subject-name** *X.500 name*—During enrollment, asks the CA to include the specified subject DN in the certificate.
- **serial-number**—During enrollment, asks the CA to include the security appliance serial number in the certificate.
- **support-user-cert-validation**—If enabled, the configuration settings to validate a remote user certificate can be taken from this trustpoint, provided that this trustpoint is authenticated to the CA that issued the remote certificate.
- **exit**—Leaves the mode.

Step 4 Save the trustpoint configuration. To do so, save the running configuration by entering the **write memory** command.

Obtaining Certificates

The security appliance needs a CA certificate for each trustpoint and one or two certificates for itself, depending upon the configuration of the keys used by the trustpoint. If the trustpoint uses separate RSA keys for signing and encryption, the security appliance needs two certificates, one for each purpose. In other key configurations, only one certificate is needed.

The security appliance supports enrollment with SCEP and with manual enrollment, which lets you paste a base-64-encoded certificate directly into the terminal. For site-to-site VPNs, you must enroll each security appliance. For remote access VPNs, you must enroll each security appliance and each remote access VPN client.

This section includes the following topics:

Obtaining Certificates with SCEP

This procedure provides steps for configuring certificates using SCEP. Repeat these steps for each trustpoint you configure for automatic enrollment. When you have completed this procedure, the security appliance will have received a CA certificate for the trustpoint and one or two certificates for signing and encryption purposes. If you use general-purpose RSA keys, the certificate received is for signing and encryption. If you use separate RSA keys for signing and encryption, the security appliance receives separate certificates for each purpose.

Note Whether a trustpoint uses SCEP for obtaining certificates is determined by the use of the **enrollment url** command when you configure the trustpoint (see the "Configuring Trustpoints" section on page 39-7).

To obtain certificates with SCEP, perform the following steps:

Step 1 Obtain the CA certificate for the trustpoint you configured.

```
hostname/contexta(config)# crypto ca authenticate trustpoint
```

For example, using trustpoint named Main, which represents a subordinate CA:

```
hostname/contexta(config)# crypto ca authenticate Main

INFO: Certificate has the following attributes:
Fingerprint:     3736ffc2 243ecf05 0c40f2fa 26820675
Do you accept this certificate? [yes/no]: y

Trustpoint 'Main' is a subordinate CA and holds a non self signed cert.
Trustpoint CA certificate accepted.
```

Step 2 Enroll the security appliance with the trustpoint. This process retrieves a certificate for signing data and, depending upon the type of keys you configured, for encrypting data.

Step 3 To perform enrollment, use the **crypto ca enroll** command. Before entering this command, contact your CA administrator because the administrator may need to authenticate your enrollment request manually before the CA grants its certificates.

```
hostname(config)# crypto ca enroll trustpoint
```

If the security appliance does not receive a certificate from the CA within 1 minute (the default) of sending a certificate request, it resends the certificate request. The security appliance continues sending a certificate request every 1 minute until a certificate is received.

Note If the fully qualified domain name configured for the trustpoint is not identical to the fully qualified domain name of the security appliance, including the case of the characters, a warning appears. If needed, you can exit the enrollment process, make any necessary corrections, and enter the **crypto ca enroll** command again.

The following enrollment example performs enrollment with the trustpoint named Main:

```
hostname(config)# crypto ca enroll Main
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
% password to the CA Administrator in order to revoke your certificate.
% For security reasons your password will not be saved in the configuration.
% Please make a note of it.
Password: 2b0rn0t2b
Re-enter password: 2b0rn0t2b
% The subject name in the certificate will be: securityappliance.example.com
% The fully-qualified domain name in the certificate will be:
securityappliance.example.com
% Include the device serial number in the subject name? [yes/no]: no
Request certificate from CA [yes/no]: yes
% Certificate request sent to Certificate authority.
```

Note The password is required if the certificate for the security appliance needs to be revoked, so it is crucial that you remember this password. Note it and store it in a safe place.

You must enter the **crypto ca enroll** command for each trustpoint with which the security appliance needs to enroll.

Note If your security appliance reboots after you issued the **crypto ca enroll** command but before you received the certificate, reissue the **crypto ca enroll** command and notify the CA administrator.

Step 4 Verify that the enrollment process was successful using the **show crypto ca certificate** command. For example, to show the certificate received from trustpoint Main:

```
hostname/contexta(config)# show crypto ca certificate Main
```

The output of this command shows the details of the certificate issued for the security appliance and the CA certificate for the trustpoint.

Step 5 Save the configuration using the **write memory** command:

```
hostname/contexta(config)# write memory
```

Obtaining Certificates Manually

This procedure provides steps for configuring certificates using manual certificate requests. Repeat these steps for each trustpoint you configure for manual enrollment. When you have completed this procedure, the security appliance will have received a CA certificate for the trustpoint and one or two certificates for signing and encryption purposes. If you use general-purpose RSA keys, the certificate received is for signing and encryption. If you use separate RSA keys for signing and encryption, the certificates received are used for each purpose exclusively.

Note Whether a trustpoint requires that you manually obtain certificates is determined by the use of the **enrollment terminal** command when you configure the trustpoint (see the "Configuring Trustpoints" section on page 39-7).

To obtain certificates manually, perform the following steps:

Step 1 Obtain a base-64 encoded CA certificate from the CA represented by the trustpoint.

Step 2 Import the CA certificate. To do so, use the **crypto ca authenticate** command. The following example shows a CA certificate request for the trustpoint Main.

```
hostname (config)# crypto ca authenticate Main
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself
MIIDRTCCAu+gAwIBAgIQKVcqP/KW74VP0NZzL+JbRTANBgkqhkiG9w0BAQUFADCB
[ certificate data omitted ]
/7QEM8izy0EOTSErKu7Nd76jwf5e4qttkQ==
quit

INFO: Certificate has the following attributes:
Fingerprint:     24b81433 409b3fd5 e5431699 8d490d34
Do you accept this certificate? [yes/no]: y
Trustpoint CA certificate accepted.

% Certificate successfully imported
hostname (config)#
```

Step 3 Generate a certificate request. To do so, use the **crypto ca enroll** command. The following example shows a certificate and encryption key request for the trustpoint Main, which is configured to use manual enrollment and general-purpose RSA keys for signing and encryption.

```
hostname (config)# crypto ca enroll Main
% Start certificate enrollment ..
```

```
% The fully-qualified domain name in the certificate will be:
securityappliance.example.com

% Include the device serial number in the subject name? [yes/no]: n

Display Certificate Request to terminal? [yes/no]: y
Certificate Request follows:

MIIBoDCCAQkCAQAwIzEhMB8GCSqGSIb3DQEJAhYSRmVyYWxQaXguY2lzY28uY29t
[ certificate request data omitted ]
jF4waw68eOxQxVmdgMWeQ+RbIOYmvt8g6hnBTrd0GdqjjVLt

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

Redisplay enrollment request? [yes/no]: n
hostname (config)#
```

Note If you use separate RSA keys for signing and encryption, the **crypto ca enroll** command displays two certificate requests, one for each key. To complete enrollment, acquire a certificate for all certificate requests generated by the **crypto ca enroll** command.

Step 4 For each request generated by the **crypto ca enroll** command, obtain a certificate from the CA represented by the applicable trustpoint. Be sure the certificate is in base-64 format.

Step 5 For each certificate you receive from the CA, use the **crypto ca import certificate** command. The security appliance prompts you to paste the certificate to the terminal in base-64 format.

Note If you use separate RSA key pairs for signing and encryption, perform this step for each certificate separately. The security appliance determines automatically whether the certificate is for the signing or encryption key pair. The order in which you import the two certificates is irrelevant.

The following example manually imports a certificate for the trustpoint Main:

```
hostname (config)# crypto ca import Main certificate
% The fully-qualified domain name in the certificate will be:
securityappliance.example.com

Enter the base 64 encoded certificate.
End with a blank line or the word "quit" on a line by itself
[ certificate data omitted ]
quit
INFO: Certificate successfully imported
hostname (config)#
```

Step 6 Verify that the enrollment process was successful using the **show crypto ca certificate** command. For example, to show the certificate received from trustpoint Main:

```
hostname/contexta(config)# show crypto ca certificate Main
```

The output of this command shows the details of the certificate issued for the security appliance and the CA certificate for the trustpoint.

Step 7 Save the configuration using the **write memory** command:

```
hostname/contexta(config)# write memory
```

Configuring CRLs for a Trustpoint

If you want to use mandatory or optional CRL checking during certificate authentication, you must perform CRL configuration for each trustpoint. For more information about CRLs, see the "About CRLs" section on page 39-3.

To configure CRLs for a trustpoint, perform the following steps:

Step 1 Enter Crypto ca trustpoint configuration mode for the trustpoint whose CRL configuration you want to modify. To do so, enter the **crypto ca trustpoint** command.

Step 2 If you have not already enabled CRLs, you can do so now by using the **crl** command with either the **required** or **optional** keyword. If you specify the **required** keyword, certificate authentication with this trustpoint cannot succeed if the CRL is unavailable.

Step 3 Enter the **crl configure** command.

```
hostname/contexta(config-ca-trustpoint)# crl configure
hostname/contexta(config-ca-crl)#
```

Upon entering this command, you enter the crl configuration mode for the current trustpoint.

Tip To set all CRL configuration options to their default values, use the **default** command. At any time while performing CRL configuration, if you want to start over, enter this command and restart this procedure.

Step 4 Configure the retrieval policy with the **policy** command. The following keywords for this command determine the policy.

- **cdp**—CRLs are retrieved only from the CRL distribution points specified in authenticated certificates.

  Note SCEP retrieval is not supported by distribution points specified in certificates.

- **static**—CRLs are retrieved only from URLs you configure.
- **both**—CRLs are retrieved from CRL distribution points specified in authenticated certificates and from URLs you configure.

Step 5 If you used the keywords static or both when you configured the CRL policy, you need to configure URLs for CRL retrieval, using the **url** command. You can enter up to 5 URLs, ranked 1 through 5.

```
hostname/contexta(config-ca-crl)# url n URL
```

where *n* is the rank assigned to the URL. To remove a URL, use the **no url** *n* command.

Step 6 Configure the retrieval method with the **protocol** command. The following keywords for this command determine the retrieval method.

- **http**—Specifies HTTP as the CRL retrieval method.
- **ldap**—Specifies LDAP as the CRL retrieval method.
- **scep**—Specifies SCEP as the CRL retrieval method.

Step 7 Configure how long the security appliance caches CRLs for the current trustpoint. To specify the number of minutes the security appliance waits before considering a CRL stale, enter the following command.

```
hostname/contexta(config-ca-crl)# cache-time n
```

where *n* is the number of minutes. For example, to specify that CRLs should be cached for seven hours, enter the following command.

```
hostname/contexta(config-ca-crl)# cache-time 420
```

Step 8 Configure whether the security appliance requires the NextUpdate field in CRLs. For more information about how the security appliance uses the NextUpdate field, see the "About CRLs" section on page 39-3.

Do one of the following:

- To require the NextUpdate field, enter the **enforcenextupdate** command. This is the default setting.
- To allow the NextUpdate field to be absent in CRLs, enter the **no enforcenextupdate** command.

Step 9 If you specified LDAP as the retrieval protocol, perform the following steps:

a. Enter the following command to identify the LDAP server to the security appliance:

```
hostname/contexta(config-ca-crl)# ldap-defaults server
```

You can specify the server by DNS hostname or by IP address. You can also provide a port number if the server listens for LDAP queries on a port other than the default of 389. For example, the following command configures the security appliance to retrieve CRLs from an LDAP server whose hostname is ldap1.

```
hostname/contexta(config-ca-crl)# ldap-defaults ldap1
```

Note If you use a hostname rather than an IP address to specify the LDAP server, be sure you have configured the security appliance to use DNS. For information about configuring DNS, see the **dns** commands in the *Cisco Security Appliance Command Reference*.

b. If LDAP server requires credentials to permit CRL retrieval, enter the following command:

```
hostname/contexta(config-ca-crl)# ldap-dn admin-DN password
```

For example:

```
hostname/contexta(config-ca-crl)# ldap-dn cn=admin,ou=devtest,o=engineering c00lRunZ
```

Step 10 To test CRL configuration for the current trustpoint, use the **crypto ca crl request** command. This command retrieves the current CRL from the CA represented by the trustpoint you specify.

Step 11 Save the running configuration. Enter the **write memory** command.

Exporting and Importing Trustpoints

You can export and import keypairs and issued certificates associated with a trustpoint configuration. The security appliance supports PKCS12 format for the export and import of trustpoints.

This section includes the following topics:

Exporting a Trustpoint Configuration

To export a trustpoint configuration with all associated keys and certificates in PKCS12 format, use the **crypto ca export** command. The security appliance displays the PKCS12 data in the terminal. You can copy the data. The trustpoint data is password protected; however, if you save the trustpoint data in a file, be sure the file is in a secure location.

The following example exports PKCS12 data for trustpoint Main using Wh0zits as the passphrase:

```
hostname (config)# crypto ca export Main pkcs12 Wh0zits

Exported pkcs12 follows:

[ PKCS12 data omitted ]

---End - This line not part of the pkcs12---

hostname (config)#
```

Importing a Trustpoint Configuration

To import the keypairs and issued certificates associated with a trustpoint configuration, use the **crypto ca import pkcs12** command in global configuration mode. The security appliance prompts you to paste the text to the terminal in base-64 format.

The key pair imported with the trustpoint is assigned a label matching the name of the trustpoint you create. For example, if an exported trustpoint used an RSA key labeled <Default-RSA-Key>, creating trustpoint named Main by importing the PKCS12 creates a key pair named Main, not <Default-RSA-Key>.

Note If a security appliance has trustpoints that share the same CA, only one of the trustpoints sharing the CA can be used to validate user certificates. The **crypto ca import pkcs12** command can create this situation. Use the **support-user-cert-validation** command to control which trustpoint sharing a CA is used for validation of user certificates issued by that CA.

The following example manually imports PKCS12 data to the trustpoint Main with the passphrase Wh0zits:

```
hostname (config)# crypto ca import Main pkcs12 Wh0zits

Enter the base 64 encoded pkcs12.
End with a blank line or the word "quit" on a line by itself:
[ PKCS12 data omitted ]
quit
INFO: Import PKCS12 operation completed successfully
hostname (config)#
```

Configuring CA Certificate Map Rules

You can configure rules based on the Issuer and Subject fields of a certificate. Using the rules you create, you can map IPSec peer certificates to tunnel groups with the **tunnel-group-map** command. The security appliance supports one CA certificate map, which can contain many rules. For more information about using CA certificate map rules with tunnel groups, see the "Creating a Certificate Group Matching Rule and Policy" section on page 27-10.

To configure a CA certificate map rule, perform the following steps:

Step 1 Enter CA certificate map configuration mode for the rule you want to configure. To do so, enter the **crypto ca certificate map command and specify the rule index number.** The following example enters CA certificate map mode for the rule with index number 1.

```
hostname(config)# crypto ca certificate map 1
hostname(config-ca-cert-map)#
```

Step 2 Use the **issuer-name** and **subject-name** commands to configure the rule. These commands specify tests that the security appliance can apply to values found in the Issuer or Subject fields of certificates. The tests can apply to specific attributes or to the whole of the Issuer or Subject fields. You can configure many tests per rule, and all the tests you specify with these commands must be true for a rule to match a certificate. Valid operators in the **issuer-name** and **subject-name** commands are as follows.

Operator	Meaning
eq	The field or attribute must be identical to the value given.
ne	The field or attribute cannot be identical to the value given.
co	Part or all of the field or attribute must match the value given.
nc	No part of the field or attribute can match the value given.

For more information about the **issuer-name** and **subject-name** commands, see the *Cisco Security Appliance Command Reference*.

The following example specifies that any attribute within the Issuer field must contain the string cisco.

```
hostname(config-ca-cert-map)# issuer-name co cisco
hostname(config-ca-cert-map)#
```

The following example specifies that within the Subject field an Organizational Unit attribute must exactly match the string Engineering.

```
hostname(config-ca-cert-map)# subject-name attr ou eq Engineering
hostname(config-ca-cert-map)#
```

Map rules appear in the output of the **show running-config** command.

```
crypto ca certificate map 1
 issuer-name co cisco
 subject-name attr ou eq Engineering
```

Step 3 When you have finished configuring the map rule, save your work. Enter the **write memory** command.