CHAPTER 2

Managing Certificates and Keys

This chapter describes how to use the import and export functions to manage the various certificate and RSA key pair files on the Cisco 4700 Series Application Control Engine (ACE) appliance. This chapter also describes the process for creating and submitting a Certificate Signing Request (CSR), which you use to obtain a certificate from a Certificate Authority (CA).

This chapter contains the following major sections:

SSL Digital Certificates and Key Pairs

Digital certificates and key pairs are a form of digital identification for user authentication. CAs, such as VeriSign and Thawte, issue certificates that attest to the validity of the public keys they contain. A client or server certificate includes the following identification attributes:

- Name of the CA (the certificate issuer) and CA digital signature
- Serial number
- Name of the client or server (the certificate subject) that the certificate authenticates
- Subject's public key
- Time stamps that indicate the certificate's expiration date

A CA has one or more signing certificates that it uses for creating SSL certificates and certificate revocation lists (CRLs). Each signing certificate has a matching private key that is used to create the CA signature. The CA makes the signing certificates (with the public key embedded) available to the public, enabling anyone to access and use the signing certificates to verify that an SSL certificate or CRL was actually signed by a specific CA.

The ACE requires certificates and corresponding key pairs for the following applications:

- SSL termination—The ACE acts as an SSL proxy server and terminates the SSL session between it and the client. For SSL termination, you must obtain a server certificate and corresponding key pair.
- SSL initiation—The ACE acts as a client and initiates the SSL session between it and the SSL server. For SSL initiation, a client certificate and corresponding key pair can be used, but is not required unless the SSL server has client authentication enabled.

Note The ACE supports 3800 certificates and 3800 key pairs. It also supports wildcard certificates.

RSA key pairs are required by an ACE and its peer during the SSL handshake in order for the two devices to establish an SSL session. The key pair refers to a public key and its corresponding private (secret) key. During the handshake, the RSA key pairs are used to encrypt the session key that both devices will use to encrypt the data that follows the handshake.

For more information on the SSL handshake process, see the "SSL Handshake" section in Chapter 1, Overview.

Before you configure the ACE for SSL termination or SSL initiation, you must import a digital certificate and its corresponding public and private key pair to the desired ACE context.

In a redundant configuration, the ACE does not synchronize the SSL certificates and key pairs that are present in the active context with the standby context of a Fault Tolerant (FT) group. If the ACE performs a configuration synchronization and does not find the necessary certificates and keys on the standby, configuration synchronization fails and the standby context enters the STANDBY_COLD state.

To copy the certificates and keys to the standby context, you can export the certificates and keys from the active context to an FTP or TFTP server using the **crypto export** command, and then import the certificates and keys to the standby context using the **crypto import** command. You can also import the certificates and keys directly to the standby context using the same method that you used to import the certificates to the active context. This second method is required if the certificates and keys were imported to the active context as non-exportable. For more information about importing and exporting certificates and keys, see the "Importing or Exporting Certificate and Key Pair Files" section.

To return the standby context to the STANDBY_HOT state in this case, you must import the necessary SSL certificates and keys to the standby context, and then perform a bulk synchronization of the active context configuration by entering the following commands in configuration mode in the active context of the FT group:

1. **no ft auto-sync running-config**
2. **ft auto-sync running-config**

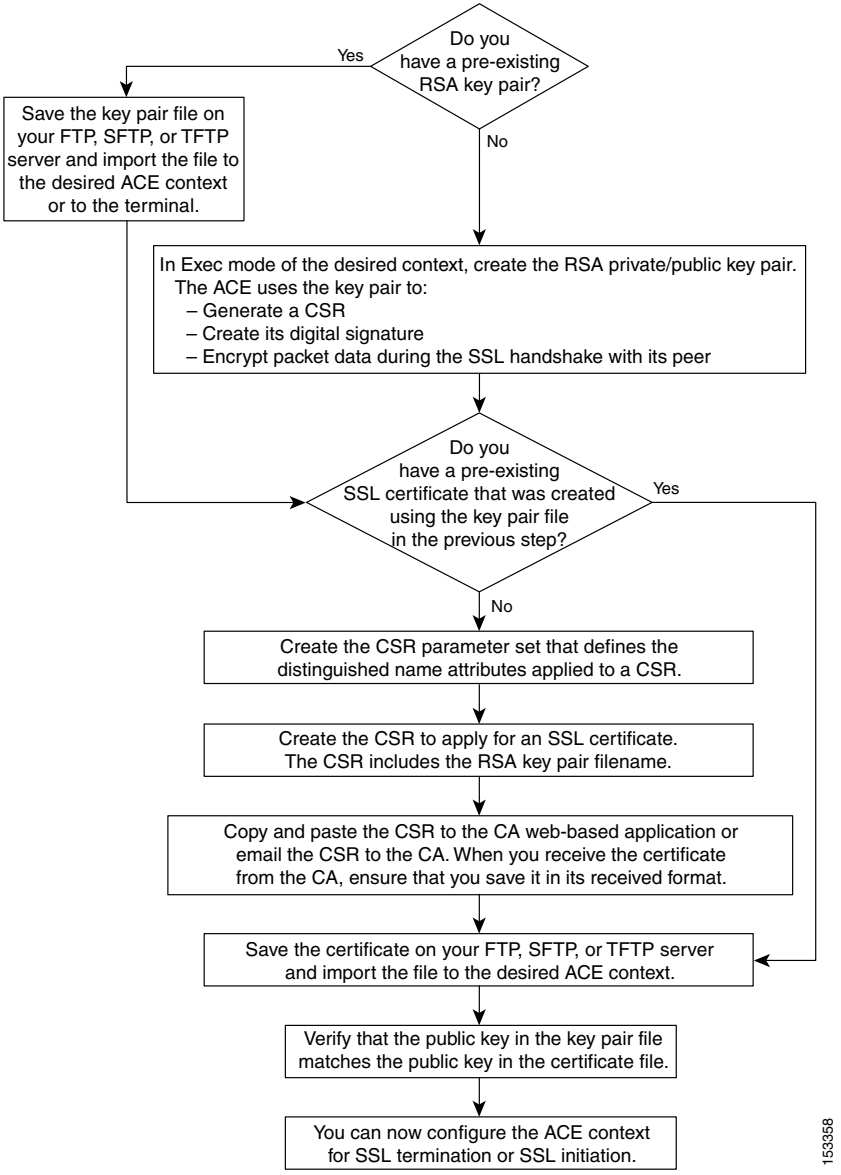
For more information about redundancy, see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*.

If you do not have a certificate and corresponding key pair, you can use the ACE to generate an RSA key pair of up to 2048 bits and a certificate signing request (CSR). You can create a CSR when you need to apply for a certificate from a CA. The CA signs the CSR and returns the authorized digital certificate to you.

Note To implement strong security policies when generating key pairs or importing certificates and key pairs, you should understand the user roles of the ACE. For more information on user roles, see the *Cisco 4700 Series Application Control Engine Appliance Virtualization Configuration Guide*.

Figure 2-1 provides an overview of how to configure an RSA key pair and SSL certificate for an ACE.

Figure 2-1 SSL Key and Certificate Configuration Overview



Generating Key Pairs and Certificate Signing Requests

If you do not have preexisting certificates and matching key pairs, the ACE includes a series of certificate and key management utilities to generate a key pair or a CSR. When the CA signs your CSR, it becomes the certificate that you can use on the ACE.

If you have preexisting certificates and matching key pairs, you can import them to the desired context on the ACE. For information on importing certificates and private keys, see the "Importing or Exporting Certificate and Key Pair Files" section.

This section contains the following topics:

Generating an RSA Key Pair

The ACE supports the generation of RSA key pairs of up to 2048 bits. To generate an RSA key pair, use the **crypto generate key** command in Exec mode.

The syntax of this command is as follows:

crypto generate key [**non-exportable**] *bitsize filename*

The arguments and keywords are as follows:

- **non-exportable**—(Optional) Specifies that the ACE marks the key pair file as nonexportable, which means that you cannot export the key pair file from the ACE.
- *bitsize*—Key pair security strength. The number of bits in the key pair file defines the size of the RSA key pair used to secure web transactions. Longer keys produce a more secure implementation by increasing the strength of the RSA security policy. Available entries (in bits) are as follows:
  - 512 (least security)
  - 768 (normal security)

  - 1024 (high security, level 1)
  - 1536 (high security, level 2)
  - 2048 (high security, level 3)

- *filename*—Name that you assign to the generated RSA key pair file. Enter an unquoted alphanumeric string with a maximum of 40 characters. The key pair filename is used only for identification purposes by the ACE.

For example, to generate the RSA key pair file MYRSAKEY.PEM, enter:

```
host1/Admin# crypto generate key non-exportable 2048 MYRSAKEY.PEM
 Generating 2048 bit RSA key pair
host1/Admin#
```

After you generate an RSA key pair, you can do the following tasks:

- Create the CSR parameter set—The CSR parameter set defines the distinguished name attributes for the ACE to use during the CSR-generating process. For details on creating a CSR configuration file, see the "Creating and Defining a CSR Parameter Set" section.
- Generate a CSR for the RSA key pair file and transfer the CSR request to the CA for signing—This action provides an added layer of security because the RSA private key originates directly within the ACE and does not have to be transported externally. Each generated key pair must be accompanied by a corresponding certificate. For details on generating a CSR, see the "Generating a Certificate Signing Request" section.

Creating and Defining a CSR Parameter Set

A CSR parameter set defines the *distinguished name* attributes that the ACE applies to the CSR during the CSR-generating process. The distinguished name attributes provide the CA with the information that it needs to authenticate your site. Creating a CSR parameter set allows you to generate multiple CSRs with the same distinguished name attributes.

Each context on the ACE can contain up to eight CSR parameter sets.

This section contains the following topics:

- Specifying a State or Province
- Specifying a Serial Number
- Specifying a Locality
- Specifying an Organization Name
- Specifying an Organizational Unit
- Specifying an E-mail Address

Creating a CSR Parameter Set

You can create a CSR parameter set by using the **crypto csr-params** command in configuration mode. You can create up to eight CSR parameter sets per context.

The syntax of this command is as follows:

crypto csr-params *csr_param_name*

The *csr_param_name* argument is the name of the CSR parameter set. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create the CSR parameter set CSR_PARAMS_1, enter:

```
host1/Admin(config)# crypto csr-params CSR_PARAMS_1
```

After you create a CSR parameter set, the CLI enters CSR parameter configuration mode, where you define the distinguished name parameters.

```
host1/Admin(config-csr-params)#
```

The distinguished name consists of several required and optional parameters. The ACE requires that you define the following CSR parameter set attributes:

- Country name
- State or province
- Common name

Note If you do not configure the required CSR parameter set attributes, the ACE displays an error message when you try to generate a CSR using the CSR parameter set.

To delete an existing CSR parameter set, enter:

```
host1/Admin(config)# no csr-params CSR_PARAMS_1
```

To display information related to existing CSR parameter sets, use the **show crypto csr-params** command (see Chapter 6, Displaying SSL Information and Statistics).

Specifying a Common Name

You can define the required common name parameter in the CSR parameter set by using the **common-name** command in CSR parameter configuration mode.

The syntax of this command is as follows:

common-name *name*

The *name* argument should be the domain name or individual hostname of the SSL site. Enter an unquoted text string with no spaces or a quoted text string with spaces, and a maximum of 64 alphanumeric characters.

For example, to specify the common name WWW.ABC123.COM, enter:

```
host1/Admin(config-csr-params)# common-name WWW.ABC123.COM
```

To delete an existing common name from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no common-name
```

Specifying a Country

You can define the required country name parameter in the CSR parameter set by using the **country** command in CSR parameter configuration mode. The syntax of this command is as follows:

country *name*

The *name* argument is the two-character code of the country where the SSL site resides (see the ISO 3166 list of country codes). Enter an unquoted text string with no spaces a maximum of two characters.

For example, to specify the country US (United States), enter:

```
host1/Admin(config-csr-params)# country US
```

To delete an existing country from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no country
```

Specifying a State or Province

You can define the required state name parameter in the CSR parameter set by using the **state** command in CSR parameter configuration mode.

The syntax of this command is as follows:

state *name*

The *name* argument is the name of the state where the SSL site resides. Enter an unquoted text string with a maximum of 40 alphanumeric characters including the ampersand (&) character and spaces.

For example, to specify the state GA (Georgia), enter:

```
host1/Admin(config-csr-params)# state GA
```

To delete an existing state from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no state
```

Specifying a Serial Number

You can define the required serial number parameter in the CSR parameter set by using the **serial-number** command in CSR parameters configuration mode.

Note The CA may choose to overwrite the serial number that you provide with their own serial number.

The syntax of this command is as follows:

serial-number *number*

The *number* argument is the serial number to assign to the certificate. Enter an unquoted text string with no spaces and a maximum of 16 alphanumeric characters including the ampersand (&) character.

For example, to specify the serial number 1001, enter:

```
host1/Admin(config-csr-params)# serial-number 1001
```

To delete an existing serial number from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no serial-number
```

Specifying a Locality

You can define the optional locality parameter in the CSR parameter set by using the **locality** command in CSR parameters configuration mode.

The syntax of this command is as follows:

locality *name*

The *name* argument is the locality name to include in the certificate. Enter an unquoted alphanumeric string with a maximum of 40 characters including spaces and the ampersand (&) character.

For example, to specify the locality ATHENS, enter:

```
host1/Admin(config-csr-params)# locality ATHENS
```

To delete an existing locality from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no locality ATHENS
```

Specifying an Organization Name

You can define the optional organization name parameter in the CSR parameter set by using the **organization-name** command in CSR parameters configuration mode. The syntax of this command is as follows:

organization-name *name*

The *name* argument is the name of the organization to include in the certificate. Enter an unquoted alphanumeric string with a maximum of 64 characters including spaces. The ACE also supports the ampersand (&) character.

For example, to specify the organization ABC123 SYSTEMS INC, enter:

```
host1/Admin(config-csr-params)# organization-name ABC123 SYSTEMS INC
```

To delete an existing organization name from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no organization-name ABC123 SYSTEMS
INC
```

Specifying an Organizational Unit

You can define the optional organization unit parameter in the CSR parameter set by using the **organization-unit** command in CSR parameters configuration mode.

The syntax of this command is as follows:

organization-unit *unit*

The *unit* argument is the name of the unit within an organization. Enter an unquoted alphanumeric string with a maximum of 64 characters including spaces and the ampersand (&) character.

For example, to specify the organization unit SSL ACCELERATOR, enter:

```
host1/Admin(config-csr-params)# organization-unit SSL ACCELERATOR
```

To delete an existing organization unit from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no organization-unit SSL ACCELERATOR
```

Specifying an E-mail Address

You can define the optional e-mail address parameter in the CSR parameter set by using the **email** command in CSR parameter configuration mode.

The syntax of this command is as follows:

email *address*

The *address* argument is the site e-mail address. Enter an unquoted alphanumeric string with no spaces and a maximum of 40 characters.

For example, to specify the e-mail address WEBADMIN@ABC123.COM, enter:

```
host1/Admin(config-csr-params)# email WEBADMIN@ABC123.COM
```

To delete an existing e-mail address from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no email
```

Generating a Certificate Signing Request

You must generate a Certificate Signing Request (CSR) file if you are requesting a new certificate or renewing a certificate. When you submit the generated CSR to a CA, the CA signs the CSR using its RSA private key and the CSR becomes the certificate.

To generate a CSR file for an RSA key pair file and to transfer the certificate request to the CA, use the **crypto generate csr** command in Exec command mode of the context containing the RSA key pair file. This command generates a CSR in PKCS10 encoded in PEM format.

The syntax of this command is as follows:

crypto generate csr *csr_params key_filename*

The arguments are as follows:

- *csr_params*—CSR parameter set that contains the distinguished name attributes (see the "Creating and Defining a CSR Parameter Set" section). Enter an unquoted alphanumeric string with no spaces and a maximum of 64 characters. The ACE applies the distinguished name attributes contained in the CSR parameter set to the CSR.
- *key_filename*—RSA key pair filename that contains the key on which the CSR is built. (This key is the public key that the ACE embeds in the CSR.) Enter an unquoted alphanumeric string with no spaces and a maximum of 40 characters. Ensure that the RSA key pair file is loaded on the ACE for the current context. If the appropriate key pair does not exist, the ACE logs an error message.

For example, to generate a CSR that is based on the CSR parameter set CSR_PARAMS_1 and the RSA key pair in the file MYRSAKEY_1.PEM, enter:

```
host1/Admin# crypto generate csr CSR_PARAMS_1 MYRSAKEY_1.PEM
-----BEGIN CERTIFICATE REQUEST-----
MIIBcDCCARoCAQAwgbQxCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlTb211U3RhdGUx
ETAPBgNVBAcTCFNvbWVDaXR5MRcwFQYDVQQKEw5BIENvbXBhbnkgTmFtZTEbMBkG
A1UECxMSV2ViIEFkbWluaXN0cmF0aW9uMR0wGwYDVQQDExR3d3cuYWNvbXBhbnlu
```

```
YW1lLmNvbTEpMCcGCSqGSIb3DQEJARYad2ViYWRtaW5AYWNvbXBhbnluYW1lLmNv
bSAwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAtBNcNXMBqh5cJHbWFsqe9LMUO90T
pYG7gF5ODvtFGREMkHh7s6S1GF131IBWCSelG4Q/qEztjCO7y3pyjruVNQIDAQAB
oAAwDQYJKoZIhvcNAQEEBQADQQCMmXRdNPBDtMQPFvylpED5UMbeaMRm2iaC+1uZ
IaHmdoX4h5eckauu9pPgSxczau8w68PF+PDS9DAAMeRDxisL
-----END CERTIFICATE REQUEST-----
(config)#
```

The **crypto generate csr** command generates the PKCS10 CSR in PEM format and outputs the CSR to the screen. Most major CAs have web-based applications that require you to cut and paste the certificate request to the screen. If necessary, you can also cut and paste the CSR to a file. The ACE does not save a copy of the CSR locally. You can, however, regenerate the same request again at any time by using the same CSR parameter set and key pair file.

Note If you require a global site certificate that allows 128-bit encryption for export-restricted browsers, you must apply for a StepUp/GSC or chained certificate from the CA. After you receive the certificate, you must prepare it for use with the ACE. For more information, see the "Preparing a Global Site Certificate" section.

After you submit your CSR to the CA, you will receive your signed certificate in one to seven business days. When you receive your certificate, you must import the certificate to the desired ACE context (see the "Importing Certificate and Key Pair Files" section).

Preparing a Global Site Certificate

Export browsers may use 40-bit encryption to initiate connections to SSL servers. With a conventional server certificate, a browser and server complete the SSL handshake and use a 40-bit key to encrypt application data.

A global site certificate is an extended server certificate that allows 128-bit encryption for export-restricted browsers. When the server responds to a browser with a global certificate, the client automatically renegotiates the connection to use 128-bit encryption.

If you applied for a global site certificate from the CA, you must obtain both the global certificate and its intermediate CA certificate. The intermediate CA certificate validates the global certificate. You can obtain a VeriSign Intermediate certificate from the following URL:

http://www.verisign.com/support/install/intermediate.html

When you receive your global site certificate and intermediate CA certificate, you must import them to the desired ACE context (see the "Importing Certificate and Key Pair Files" section). Then you create a certificate chain group that includes both certificates (see the "Creating a Chain Group" section). The ACE sends the chain group to the client during the initial SSL handshake.

Importing or Exporting Certificate and Key Pair Files

You can import PEM-encoded certificate and key pair files to the ACE from a remote secure server. To transfer these files, we recommend that you use a secure encrypted transport mechanism between the ACE and the remote server.

The ACE supports the Secure Shell (SSHv2) protocol, which provides secure encryption communications between two hosts over an insecure network. For file transport between network devices, the ACE supports Secure File Transfer Protocol (SFTP), File Transfer Protocol (FTP), and Trivial File Transfer Protocol (TFTP). We recommend that you use SFTP because it is the only one of these three protocols that can provide a secure and encrypted connection.

Before you import a certificate or key pair file to the ACE, you must perform the following tasks:

- On the ACE, ensure that SSH access to the ACE is enabled to accept connections from SSH clients. By default, SSH access is enabled. If you restrict SSH access, the ACE will not accept connections from SSH clients and the import command will fail (an error message will be generated).

Note For details about configuring the Secure Shell daemon on the Catalyst 6500 series switch or Cisco 7600 series router, see the *Catalyst 6500 Series Switch Cisco IOS Software Configuration Guide* or *Cisco 7600 Series Router Cisco IOS Software Configuration Guide*.

- On the SFTP server, verify that the server is properly configured. The user directory must point to the directory where the certificates and key pairs reside. This path is required by the ACE to ensure that certificates and keys are properly copied from or to the SFTP server.

This section contains the following topics:

Importing Certificate and Key Pair Files

The ACE supports the importation of PEM-encoded key pairs and certificates (including wildcard certificates) signed by keys of up to and including 2048 bits. You can import a certificate or key pair file to the ACE from a remote server by using the **crypto import** command in Exec mode. You can import individual certificates and keys. Because a network device uses its certificate and corresponding public key together to prove its identity during the SSL handshake, be sure to import both the certificate file and its corresponding key pair file.

Note If you attempt to import a file that has the same filename of an existing local file, the ACE appliance does not overwrite the existing file. Before importing the updated file, you must either delete the local file or rename the imported file. For more information, see the "Deleting Certificate and Key Pair Files" or "Upgrading an SSL Certificate" section.

The syntax of this command is as follows:

crypto import [**non-exportable**] {{**ftp** | **sftp**} [**passphrase** *passphrase*] *ip_addr username remote_filename local_filename*} | {**tftp** [**passphrase** *passphrase*] *ip_addr remote_filename local_filename*} | **terminal** *local_filename* [**passphrase** *passphrase*]

The keywords, arguments, and options are as follows:

- **non-exportable**—(Optional) Marks the imported file as nonexportable, which means that you cannot export the file from the ACE.
- **ftp**—Specifies the File Transfer Protocol file transfer process.
- **sftp**—Specifies the Secure File Transfer Protocol file transfer process.

- **tftp**—Specifies the Trivial File Transfer Protocol file transfer process.
- **passphrase** *passphrase*—(Optional) Indicates that the file was created with a passphrase, which you must submit with the file transfer request in order to use the file. The passphrase can contain a maximum of 39 characters and pertains only to encrypted PEM files and PKCS files.
- *ip_addr*—IP address of the remote server. Enter an IP address in dotted-decimal notation (for example, 192.168.12.15).
- *username*—Username required to access the remote server. When you execute the command, the ACE prompts you for the password of the username on the remote server.
- *remote_filename*—Name of the certificate or key pair file that resides on the remote server to import.
- *local_filename*—Name to save the file to when imported to the ACE. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- **terminal**—Allows you to import a file using cut and paste by pasting the certificate and key pair information to the terminal display. You must use the terminal method to display PEM files, which are in ASCII format.

The ACE supports the importation of PEM-encoded SSL certificates and keys with a maximum line width of 130 characters using the terminal. If an SSL certificate or key is not wrapped or it exceeds 130 characters per line, use a text editor such as the visual (vi) editor or Notepad to manually wrap the certificate or key to less than 130 characters per line. Alternatively, you can import the certificate or key by using SFTP, FTP, or TFTP with no regard to line width. Of these methods, we recommend SFTP because it is secure.

For example, to
import the RSA key file MYRSAKEY.PEM from an SFTP server, enter:

```
host1/Admin# crypto import non-exportable sftp 1.1.1.1 JOESMITH
/USR/KEYS/MYRSAKEY.PEM MYKEY.PEM
Password: ********
Passive mode on.
Hash mark printing on (1024 bytes/hash mark).
#
Successfully imported file from remote server.
host1/Admin#
```

The following example shows how to use the **terminal** keyword to allow pasting of the certificate information to the file MYCERT.PEM:

```
host1/Admin# crypto import terminal MYCERT.PEM
Enter PEM formatted data ending with a blank line or "quit" on a line
by itself
--------BEGIN CERTIFICATE-----------------------
MIIC1DCCAj2gAwIBAgIDCCQAMA0GCSqGSIb3DQEBAgUAMIHEMQswCQYDVQQGEwJa
QTEVMBMGA1UECBMMV2VzdGVybiBDYXBlMRIwEAYDVQQHEwlDYXBlIFRvd24xHTAb
BgNVBAoTFFRoYXd0ZSBDb25zdWx0aW5nIGNjMSgwJgYDVQQLEx9DZXJ0aWZpY2F0
aW9uIFNlcnZpY2VzIERpdmlzaW9uMRkwFwYDVQQDExBUaGF3dGUgU2VydmVyIENB
MSYwJAYJKoZIhvcNAQkBFhdzZXJ2ZXItY2VydHNAdGhhd3RlLmNvbTAeFw0wMTA3
-----------END CERTIFICATE-----------------------
quit
```

Exporting Certificate and Key Pair Files

You can export a certificate or key pair file from the ACE to a remote server or the terminal screen by using the **crypto export** command in Exec command mode.

Note You cannot export a certificate or key pair file that you marked as nonexportable when you imported the file to the ACE.

The syntax of this command is as follows:

crypto export *local_filename* {**ftp** | **sftp** | **tftp** | **terminal**} {*ip_addr*} {*username*} {*remote_filename*}

The keywords, arguments, and options are as follows:

- *local_filename*—Name of the file that resides on the ACE to export. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- **ftp**—Specifies the File Transfer Protocol file transfer process.
- **sftp**—Specifies the Secure File Transfer Protocol file transfer process. We recommend that you use SFTP because it is more secure than FTP or TFTP.
- **tftp**—Specifies the TFTP Trivial File Transfer Protocol file transfer process.
- **terminal**—Displays the file content on the terminal for copy and paste purposes. Use the **terminal** keyword when you need to cut and paste certificate or private key information from the console. You must use the terminal method to display PEM files, which are in ASCII format.

- *ip_addr*—IP address or name of the remote server. Enter an IP address in dotted-decimal notation (for example, 192.168.12.15).
- *username*—Username required to access the remote server. The ACE prompts you for your password when you execute the command.
- *remote_filename*—Name to save the file to on the remote server.

The remote server variables listed after the **terminal** keyword are used by the ACE when you select a transport type of **ftp**, **sftp**, or **tftp** (the variables are not used for **terminal**). If you select one of these transport types and do not define the remote server variables, the ACE prompts you for the variable information.

For example, to use SFTP to export the key file MYKEY.PEM from the ACE to a remote SFTP server, enter:

```
host1/Admin# crypto export MYKEY.PEM sftp 192.168.1.2 JOESMITH
/USR/KEYS/MYKEY.PEM
User password: ****
Writing remote file /usr/keys/mykey.pem
host1/Admin#
```

Upgrading an SSL Certificate

To upgrade an SSL certificate without disrupting active SSL sessions or pending SSL sessions, use the following steps:

Step 1 Import the new SSL certificate using the **crypto import** command in Exec mode and save it with a new name. See the "Importing Certificate and Key Pair Files" section. For example, to import a certificate from an SFTP server, enter the following command:

```
host1/Admin# crypto import non-exportable sftp 1.1.1.1 JOESMITH
/USR/CERTS/MY_CERT.PEM MY_NEW_CERT.PEM
Password: ********
Passive mode on.
Hash mark printing on (1024 bytes/hash mark).
#
Successfully imported file from remote server.
host1/Admin#
```

Step 2 While the SSL proxy service is actively processing flows, change the certificate file association within the SSL proxy service to the new certificate by using the **cert** command in SSL proxy configuration mode. See the "Specifying the Certificate" section in Chapter 3, Configuring SSL Termination.

For example, to associate the certificate in the MY_NEW_CERT.PEM certificate file with the PSERVICE_SERVER SSL proxy service, enter the following commands:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
host1/Admin(config-ssl-proxy)# cert MY_NEW_CERT.PEM
```

Verifying a Certificate Against a Key Pair

A digital certificate is built around the public key of a key pair and can only be used with one key pair. You can compare the public key in a certificate file with the public key in a key pair file and verify that they are identical by using the **crypto verify** command in Exec command mode.

Note If the public key in the certificate does not match the public key in the key pair file, the ACE logs an error message.

The syntax of this command is as follows:

crypto verify *key_filename cert_filename*

The arguments are as follows:

- *key_filename*—Name of the context key pair file that the ACE uses to verify against the specified certificate. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- *cert_filename*—Name of the context certificate file that the ACE uses to verify against the specified key pair. Enter an unquoted alphanumeric string with a maximum of 40 characters.

For example, to verify the public keys in the files MYRSAKEY.PEM and MYCERT.PEM match, enter:

```
host1/Admin# crypto verify myrsakey.pem mycert.pem
keypair in myrsakey.pem matches certificate in mycert.pem
```

The following example shows what the ACE displays when the public keys do not match:

```
host1/Admin# crypto verify myrsakey_2.pem mycert.pem
Keypair in myrsakey_2.pem does not match certificate in mycert.pem
host1/Admin#
```

Deleting Certificate and Key Pair Files

You can delete certificate and key pair files that are no longer valid by using the **crypto delete** command in Exec command mode. Because the ACE appliance does not overwrite existing certificate or key pair files, deleting the file allows you to import an updated file.

The syntax of this command is as follows:

crypto delete {*filename* | **all**}

The keywords and arguments are as follows:

- *filename*—Name of a specific certificate or key pair file to delete. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- **all**—Deletes all certificate and key pair files from the context.

To display a list of available certificate and key pair files loaded on the ACE, use the **show crypto files** command.

Note The **crypto delete** command deletes the specified context crypto files from flash memory; however, existing SSL services are not interrupted. If you do not replace the deleted SSL files, the SSL services are disabled the next time that you enter the **vip inservice** command or when a device reload occurs.

For example, to delete the key pair file MYRSAKEY.PEM, enter:

```
host1/Admin# crypto delete MYRASKEY.PEM
```

Creating a Chain Group

A chain groups specifies the *certificate chains* that the ACE sends to its peer during the handshake. A certificate chain is a hierarchal list of certificates that includes the subject's certificate, the root CA certificate, and any intermediate CA certificates. Using the information provided in a certificate chain, the certificate verifier can search for a trusted authority in the certificate hierarchal list back to the root CA. The verifier may find what it considers a trusted authority before reaching the root CA certificate, in which case, the verifier stops searching.

When defining an SSL proxy service, you can configure the service with a chain group (see the "Creating and Defining an SSL Proxy Service" section in Chapter 3, Configuring SSL Termination).

The ACE supports the following certificate chain group capabilities:

- A chain group can contain up to eight certificate chains.
- Each context on the ACE can contain up to eight chain groups.
- The maximum size of a chain group is 16 KB.

To create a chain group, use the **crypto chaingroup** command in configuration mode.

The syntax of this command is as follows:

crypto chaingroup *group_name*

The *group_name* argument is the name of the chain group. Enter an unquoted alphanumeric string with no spaces and a maximum of 64 characters.

For example, to create the chain group MYCHAINGROUP, enter:

```
host1/Admin(config)# crypto chaingroup MYCHAINGROUP
```

After you create a chain group, the CLI enters chaingroup configuration mode, where you add the required certificate files to the group.

```
host1/Admin(config-chaingroup)#
```

To delete an existing chain group, enter:

```
host1/Admin(config)# no crypto chaingroup MYCHAINGROUP
```

You can add certificate files to the chain group by using the **cert** command in chaingroup configuration mode. You can configure a chaingroup with up to nine certificates.

The syntax of this command is as follows:

cert *cert_filename*

The *cert_filename* argument is the name of an existing certificate file stored on the ACE. Enter an unquoted alphanumeric string with a maximum of 40 characters.

Note When you make a change to a chain-group certificate, the change takes effect only after you respecify the associated chain group in the SSL proxy service using the **chaingroup** command. See the "Creating and Defining an SSL Proxy Service" section in Chapter 3, Configuring SSL Termination.

Typically, it is not necessary to add the certificates to the chain group in any type of hierarchical order because the device that verifies the certificates determines the correct order. However, some mobile devices may not be able to order the certificates properly and will display an error message. In this case, you need to add the certificates to the chain group in the correct order.

To display a list of existing certificate files, use the **show crypto files** command (see the "Displaying Certificate Information" section in Chapter 6, Displaying SSL Information and Statistics).

For example, to add the certificate files MYCERTS.PEM and MYCERTS_2.PEM to the chain group, enter:

```
host1/Admin(config-chaingroup)# cert MYCERTS.PEM
host1/Admin(config-chaingroup)# cert MYCERTS_2.PEM
```

To remove a certificate file from the chain group, enter:

```
host1/Admin(config-chaingroup)# no cert MYCERTS_2.PEM
```

Configuring a Group of Certificates for Authentication

On the ACE, you can implement a group of four SSL certificates that are trusted as certificate signers by creating an authentication group. After creating the authentication group and assigning its certificates, you can assign the authentication group to a service in an SSL termination configuration to enable client authentication. For information on client authentication, see the "Enabling Client Authentication" section in Chapter 3, Configuring SSL Termination.

You can also assign the group to a service in an SSL initiation configuration to allow the ACE to authenticate the server certificate with the group certificates. For information on server authentication, see the "Configuring an Authentication Group for Server Authentication" section in Chapter 4, Configuring SSL Initiation.

To create an authentication group and access authgroup configuration mode, use the **crypto authgroup** command in configuration mode. The syntax of this command is as follows:

crypto authgroup *group_name*

The *group_name* argument is the name of the certificate authentication group. Enter an unquoted alphanumeric string with no spaces and a maximum of 64 characters.

For example, to create the authentication group AUTH-CERT1, enter:

```
host1/Admin(config)# crypto authgroup AUTH-CERT1
```

After you create an authentication group, you access authgroup configuration mode, where you add the required certificate files to the group.

```
host1/Admin(config-authgroup)#
```

To delete an existing authentication group, enter:

```
host1/Admin(config)# no crypto authgroup AUTH-CERT1
```

To add certificate files to the authentication group, use the **cert** command in authgroup configuration mode. You can configure an authentication group with up to four certificates.

The syntax of this command is as follows:

cert *cert_filename*

The *cert_filename* argument is the name of an existing certificate file stored on the ACE. Enter an unquoted alphanumeric string with a maximum of 40 characters.

Note When you make a change to an authgroup, the change takes effect immediately.

To display a list of existing certificate files, use the **show crypto files** command (see the "Displaying Certificate Information" section in Chapter 6, Displaying SSL Information and Statistics). It is not necessary to add the certificates in any type of hierarchal order because the device verifying the certificates determines the correct order.

For example, to add the certificate files MYCERTS.PEM and MYCERTS_2.PEM to the authentication group, enter:

```
host1/Admin(config-authgroup)# cert MYCERTS.PEM
host1/Admin(config-authgroup)# cert MYCERTS_2.PEM
```

To remove a certificate file from the authentication group, enter:

```
host1/Admin(config-authgroup)# no cert MYCERTS_2.PEM
```