Configuring SSL Certificates and Keys

This chapter describes how to generate and import SSL certificates and keys, and how to associate them with files for use in the CSS. It contains the following major sections:

Overview of SSL Certificates and Keys

Digital certificates and key pairs are a form of digital identification for user authentication. Certificate Authorities (CAs), such as VeriSign and Thawte, issue certificates. A client or server certificate includes the name of the issuing authority and digital signature, the serial number, the name of the client or server that the certificate was issued for, the public key, and the time stamps that indicate the certificate's expiration date.

A CA also provides a trusted CA certificate to verify that a client or server certificate originated from the CA. This certificate also can verify that a certificate revocation list (CRL) originated from the CA. This CA certificate includes the CA distinguished name, public key, and digital signature.

The CSS require certificates and keys for:

- SSL termination - You must obtain a server certificate and key.
- SSL initiation - You must obtain a client certificate and key.
- Client authentication - You must obtain a trusted CA certificate from the CA to verify that the client certificate and certificate revocation list (CRL) were issued by the CA.

Before configuring SSL termination, client authentication, or SSL initiation, you must load a digital certificate on the CSS disk (flash disk or hard disk). For SSL termination or SSL initiation, you must also load a public/private key pair on the CSS. The CSS stores digital certificates and key pairs in encrypted files in a secure area on the CSS.

For server and client certificates, you can use files received from a CA, import the certificate and keys from an existing secure server, or generate your own certificate and keys on the CSS. The CSS supports the generation of certificates and keys directly within the CSS for purposes of testing. Your requirement to use generated certificates and keys instead of certificates and keys from a trusted authority depends on your environment. For example, the use of the CSS and SSL for a company's internal website may not require the use of certificates from a trusted CA. A certificate and key pair generated within the CSS may be sufficient to satisfy the intranet SSL requirement.

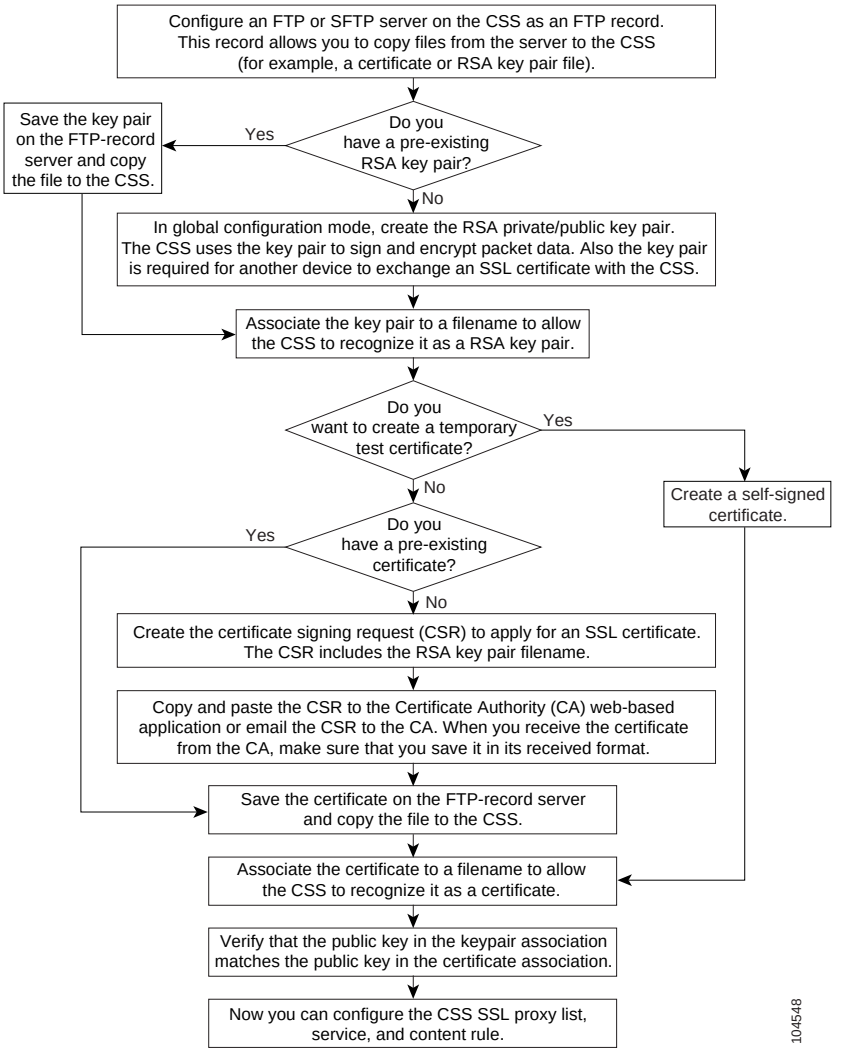
After you import certificates or key pairs on the CSS, you must associate them to filenames. You will use these filenames when you configure SSL termination, client authentication, or SSL initiation.

Caution When importing or exporting certificates and keys with the CSS, ensure that the CSS is not configured to perform a network boot from a network-mounted file system on a remote system (operating the CSS in a diskless environment). The network-mounted method of CSS booting is not supported with SSL termination; the certificates and keys must be local to the CSS and SSL module.

Note To implement good security policies when importing or generating SSL certificates and key pairs, administrators should understand the user modes of the CSS and have strong password policies to protect those user modes. For more information, refer to the *Cisco Content Services Switch Command Reference*, Chapter 2, CLI Commands, the "(config) username-technician" section.

Figure 3-1 provides an overview of how to configure an RSA key pair and SSL server certificate on the CSS.

Figure 3-1 SSL Key and Server Certificate Configuration Overview



Generating Certificates and Private Keys in the CSS

If you have preexisting certificates and private keys, you can import them to the CSS disk. For information on importing preexisting certificates and private keys, see the "Importing or Exporting Certificates and Private Keys" section.

If you do not have preexisting keys, Diffie-Hellman parameters, and certificates for the CSS, the CSS includes a series of certificate and private key management utilities to generate them. These utilities simplify the process of generating an RSA private key, a DSA private key, a Diffie-Hellman parameter file, a certificate signing request (CSR), and a self-signed temporary certificate.

Note The **ssl genrsa**, **gencsr**, **gendsa**, and **gencert** commands all produce a valid certificate or key pair. Be aware, however, that most Web browsers will flag the certificate as signed by an unrecognized signing authority.
A generated certificate is temporary and expires in one year. The **ssl gencsr** command generates a certificate request in PKCS10 encoded in Privacy Enhanced Mail (PEM) format.

This section covers:

Generating an RSA Key Pair

RSA key pairs are used to sign and encrypt packet data, and they are required before another device (client or server) can exchange an SSL certificate with the CSS. The key pair refers to a public key and its corresponding private (secret) key. The CSS stores the generated RSA key pair as a file on the CSS.

Use the **ssl genrsa** command to generate an RSA private/public key pair for asymmetric encryption. The syntax for this command is:

ssl genrsa *filename numbits* **"***password***"**

The variables are:

- *filename* - The name of generated RSA key pair file. Enter an unquoted text string with a maximum of 31 characters. The key pair filename is used only for identification in the CSS.
- *numbits* - The key pair 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 512 (least security), 768 (normal security), 1024 (high security), and 2048 (highest security).
- **"***password***"** - The password used to encode the RSA private key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the RSA key pair *myrsakeyfile1*, enter:

```
(config) # ssl genrsa myrsakeyfile1 1024 "passwd123"
 Please be patient this could take a few minutes
```

After you generate an RSA key pair, you can generate a Certificate Signing Request (CSR) file for the RSA key pair file and transfer the certificate request to the Certificate Authority (CA). This provide an added layer of security because the RSA private key originates directly within the CSS and does not have to be transported externally. You can then create a temporary certificate for internal testing until the CA responds to the certificate request and returns the authentic certificate. Each generated key pair must be accompanied by a certificate to work.

You must also associate an RSA key pair name with the generated RSA key pair, as discussed in the "Associating Certificate and Private Key Files with Names" section of this chapter.

Generating a DSA Key Pair

DSA is the public key exchange cryptographic system developed by the National Institutes of Science and Technology. DSA can only be used for digital signatures (signings); it cannot be used for key private/public exchange. The CSS stores the generated DSA key pair as a file on the CSS.

Use the **ssl gendsa** command to generate a DSA private/public key pair for asymmetric encryption. The syntax for this command is:

ssl gendsa *filename numbits* **"***password***"**

The variables are:

- *filename* - The name of the generated DSA key pair file. Enter an unquoted text string with a maximum of 31 characters. The key pair filename is used only for identification in the CSS.
- *numbits* - The key pair strength. The number of bits in the key pair file defines the size of the DSA key pair used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the DSA security policy. Available entries (in bits) are 512 (least security), 768 (normal security), and 1024 (highest security).
- **"***password***"** - The password used to encode the DSA private key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the DSA key pair *mydsakeyfile2*, enter:

```
(config) # ssl gendsa mydsakeyfile2 512 "passwd123"
Please be patient this could take a few minutes
```

You must also associate a DSA key pair name with the generated DSA key pair as discussed in the "Associating Certificate and Private Key Files with Names" section of this chapter.

Generating Diffie-Hellman Key Parameters

Diffie-Hellman is a shared key agreement algorithm. Diffie-Hellman key exchange uses a complex algorithm and public/private keys to encrypt and then decrypt packet data. The CSS stores the generated Diffie-Hellman key parameter file. Use the **ssl gendh** command to generate a Diffie-Hellman key agreement parameter file.

Note Generation of a Diffie-Hellman key agreement parameter file can sometimes take a lengthy period of time (perhaps up to 20 minutes) and is a CPU-intensive utility. If you are running the **ssl gendh** utility, ensure that the CSS is not actively passing traffic at the same time to avoid impacting CSS performance.

The syntax for this command is:

ssl gendh *filename numbits* **"***password***"**

The variables are:

- *filename* - The name of the file to store the Diffie-Hellman key parameters. Enter an unquoted text string with a maximum of 31 characters. The filename is used only for identification in the CSS.
- *numbits* - The key strength. The number of bits in the file defines the size of the Diffie-Hellman key used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the Diffie-Hellman security policy. Available entries (in bits) are 512 (least security), 768 (normal security), 1024 (high security), and 2048 (highest security).
- **"***password***"** - The password used to encode the Diffie-Hellman key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the Diffie-Hellman key parameter list *dhparamfile2*, enter:

```
(config) # ssl gendh dhparamfile2 512 "passwd123"
Please be patient this could take a few minutes
```

You must also associate a Diffie-Hellman parameter filename with the generated Diffie-Hellman parameter file, as discussed in the "Associating Certificate and Private Key Files with Names" section of this chapter.

Using an RSA Key to Generate a Certificate Signing Request

To generate a Certificate Signing Request (CSR) file for an RSA key pair file and to transfer the certificate request to the Certificate Authority (CA), use the **ssl gencsr** *rsakey* command. This command generates a CSR in PKCS10 encoded in PEM format.

You must generate a CSR file if you are requesting a new certificate or renewing a certificate. When the CA signs the CSR using its RSA private key, the CSR becomes the certificate.

The *rsakey* variable specifies the key on which the RSA certificate is built. It is the public key that is embedded in the certificate.

To use the RSA key pair to generate a CSR, ensure the RSA key pair file is loaded on the CSS. Associate an RSA key pair name to the generated RSA keypair (see the "Associating Certificate and Private Key Files with Names" section). If the appropriate key pair does not exist, the CSS logs an error message.

For example, to generate a CSR based on the RSA key pair *myrsakey1*, enter:

```
CSS11503(config)# ssl gencsr myrsakey1
You are about to be asked to enter information
that will be incorporated into your certificate
request. What you are about to enter is what is
called a Distinguished Name or a DN.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [US]US
State or Province (full name) [SomeState]New York
Locality Name (city) [SomeCity]Albany
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.
Organizational Unit Name (section) [Web Administration]Web Admin
Common Name (your domain name) [www.acme.com]www.cisco.com
Email address [webadmin@acme.com]webadmin@cisco.com

-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQICAQAwgZwxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJNQTETMBEGA1UE
BxMKQm94Ym9yb3VnaDEcMBoGA1UEChMTQ2lzY28gU3lzdGVtcywgSW5jLjESMBAG
A1UECxMJV2ViIEFkbWluMRYwFAYDVQQDEw13d3cuY2lzY28uY29tMSEwHwYJKoZI
hvcNAQkBFhJra3JvZWJlckBjaXNjby5jb20wXDANBgkqhkiG9w0BAQEFAANLADBI
```

```
AkEAqHXjtQUVXvmo6tAWPiMpe6oYhZbJUDgTxbW4VMCygzGZn2wUJTgLrifDB6N3
v+1tKFndE686BhKqfyOidml3wQIDAQABoAAwDQYJKoZIhvcNAQEEBQADQQA94yC3
4SUJJ4UQEnO2OqRGLOZpAElc4+IV9aTWK6NmiZsM9Gt0vPhIkLx5jjhVRLlb27Ak
H6D5omXa0SPJan5x
-----END CERTIFICATE REQUEST-----

CSS11503(config)#
```

The **ssl gencsr** command generates the CSR in PKCS10 encoded in PEM format and outputs it to the screen. Most major Certificate Authorities 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 certificate to a file. Note that the CSR is not saved in the CSS.

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

After submitting your CSR to the Certificate Authority (CA), you will receive your signed certificate between one to seven business days. When you receive your CSR, import the CSR to the CSS and then associate it. For information on importing the CSR, see the "Importing or Exporting Certificates and Private Keys" section. For information on associating it, see the "Associating Certificate and Private Key Files with Names" section.

While you are waiting to receive your signed certificate, you can test your CSR file by creating a temporary certificate by generating a CSR and signing it with your own private key. While this produces a valid certificate, most browsers flag the certificate as signed by an unrecognized signing authority. To generate a temporary certificate, see the "Generating a Self-Signed Certificate" section.

Generating a Self-Signed Certificate

For purposes of SSL testing, you can generate a temporary certificate by generating a CSR and signing it with your own private key. A generated certificate is temporary and expires in 30 days. Use the **ssl gencert** command to generate and save a temporary certificate to a file on disk in the CSS.

Note The **ssl gencert** command produces a valid certificate. However, most Web browsers flag this certificate as signed by an unrecognized signing authority.

Before you generate the certificate, consider:

- The key pair that the certificate is based on (RSA or DSA).
- The key used to sign the certificate.

The **ssl gencert** command can sign RSA or DSA certificates with either an RSA key pair or a DSA key pair.

Note Although the CSS allows signing an RSA certificate with a DSA key (and a DSA certificate with an RSA key) it is a more standard practice that an RSA certificate is signed with RSA keys (and DSA certificate is signed with a DSA key).

The syntax for this command is:

ssl gencert certkey *certkey* **signkey** *signkey certfile* **"***password***"**

The variables are:

- **certkey** *certkey* - The name of the RSA or DSA key pair on which the certificate is based. Enter an unquoted text string with a maximum of 31 characters.
- **signkey** *signkey* - The RSA or DSA key pair to be used to sign the certificate. Enter an unquoted text string with a maximum of 31 characters.
- *certfile* - The name of the file used to store the certificate as a file on the CSS. Enter an unquoted text string with a maximum of 31 characters.
- **"***password***"** - The password used to encode the certificate file using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private

key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to interactively generate the *mycertfile2* certificate, enter:

```
CSS11503(config)# ssl gencert certkey myrsakey signkey myrsasignkey
myrsacertfile "passwd123"
You are about to be asked to enter information
that will be incorporated into your certificate
request. What you are about to enter is what is
called a Distinguished Name or a DN.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [US]US
State or Province (full name) [SomeState]New York
Locality Name (city) [SomeCity]Albany
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.
Organizational Unit Name (section) [Web Administration]Web Admin
Common Name (your domain name) [www.acme.com]www.cisco.com
Email address [webadmin@acme.com]webadm@cisco.com

CSS11503(config)#
```

You must also associate the contents of this temporary certificate to a filename, as discussed in the "Associating Certificate and Private Key Files with Names" section of this chapter.

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

Then you must chain both certificates together in a single file, creating a chained certificate. As one file, the CSS returns the entire certificate chain to the client upon the initial SSL handshake.

Copy the server global and intermediate certificates to an FTP server. When creating a chained certificate for the CSS, make sure that the global and intermediate certificate are in their proper order. In one file, paste the server global site certificate first, followed by the intermediate certificate. You must insert a single new line between the certificates.

Save the file and import it to the CSS, as described in the "Importing or Exporting Certificates and Private Keys" section.

Importing or Exporting Certificates and Private Keys

You can import preexisting or new certificates and private keys to the CSS disk from a file, or a series of files, that are stored on a remote secure server. For information on generating certificates, see the "Generating Certificates and Private Keys in the CSS" section.

To transfer these files, we recommend that you use a secure encrypted transport mechanism between the CSS and the remote server. The CSS supports the Secure Shell protocol (SSHv2), which provides secure encryption communications between two hosts over an insecure network. The CSS supports file transport between network devices using the Secure File Transfer Protocol (SFTP) and the File Transfer Protocol (FTP). Of the two file transport protocols, we recommend SFTP as the transport mechanism of choice. It is similar to FTP except that it uses a secure and encrypted connection.

Before you import certificates or keys to the CSS:

- On the CSS, ensure that SSH access to the CSS is enabled to accept connections from SSH clients and that the Secure Management license key is installed prior to transferring certificates and keys. By default, SSH access is enabled through the **no restrict ssh** global command. If SSH access is restricted, or if the license key is not installed, SSH will not accept connections from SSH clients and the **copy ssl sftp** command will fail, resulting in generation of an error message.

Note For details about configuring Secure Shell Daemon on the CSS, refer to the *Cisco Content Services Switch Security Configuration Guide*.

- On the SFTP server, verify that the server is properly configured so that the user directory points to the directory where the certificates and keys reside. This path is required to ensure certificates and keys are properly copied from or to the SFTP server.

Caution When using SSH, ensure that the CSS is not configured to perform a network boot from a network-mounted file system on a remote system (a diskless environment). If SSH is enabled and the CSS has been booted using a network boot from a network-mounted file system, the CSS logs an error message by SSH as the protocol attempts to initialize and then exits from operation, which impacts importing and exporting certificates and keys.

Configuring the Default SFTP or FTP Server to Import Certificates and Private Keys

Before you begin, use the **ftp-record** command to define the SFTP or FTP server that you intend to use to download imported certificates and private keys to the CSS disk. For details about using the **ftp-record** command to create an SFTP or FTP record file to use when accessing the server from the CSS, refer to the *Cisco Content Services Switch Administration Guid*e.

Note When defining the FTP record for the **copy ssl** command, ensure that the base directory, if used, is relative to the SSH directory where the SSH server resides. For example, if the username is *sshlogin* and the SSH server is installed in d:\Program Files\Network, the default directory for the files would be d:\Program Files\Network\ssh. This path is required to ensure certificates and keys are properly copied to or from the SFTP server.

For example, to define the *ssl_record*, enter:

```
# ftp-record ssl_record 192.168.19.21 johndoe "abc123" /home/johndoe
```

Transferring Certificates and Private Keys to the CSS

To facilitate the import or export of certificates and private keys from or to the CSS, use the **copy ssl** command. The CSS stores all imported files in a secure location on the CSS. This command is available only in SuperUser mode.

The syntax for this command is:

copy ssl [*protocol*] *ftp_record* [**import** *filename* [*format*] **"***password***"** {**"***passphrase***"**}|**export** *filename2* **"***password***"**]

The variables are:

- *protocol* - The type of protocol used to transfer the certificate and private key file. The valid entries are **sftp** or **ftp**. We recommend the SFTP protocol for the transport mechanism because it provides the most security.
- *ftp_record* - The name of the previously-created FTP record containing the remote host information.
- **import** - Imports the file from the remote server.
- *filename* - The name of the file you want to import from the server. Include the full path to the file. You can enter a maximum of 128 characters.
- *format* - The file format of the certificate to be imported. Once the certificate file is converted to PEM format and DES encoded, it is stored on the CSS SCM in a special (and secure) directory. The valid import file formats are:
  - **DER** - Binary format encoding of the certificate file in ASN.1 using the Distinguished Encoding Rules (DER-encoded X509 certificate). For example, an imported certificate from a Microsoft Windows NT IIS 4.0 server.
  - **PEM** - Privacy Enhanced Mail, a base64 encoding of the certificate file (PEM-encoded X509 certificate). For example, an imported certificate from an Apache/SSL UNIX server.
  - **PKCS12** - Standard from RSA Data Security, Inc. for storing certificates and private keys. For example, an imported certificate from a Microsoft Windows 2000 IIS 5.0 server.

- *"password"* - The password used to DES (Data Encryption Standard) encode the imported certificate or private key. Encoding the imported file prevents unauthorized access to the certificate or private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.
- *"passphrase"* - (Optional for PEM files) The passphrase used to encrypt the certificate or key being imported into the CSS. Enter the passphrase as a quoted text string.

Note You must enter a passphrase for a PKCS12 file (.pfx). The CSS uses the passphrase to decrypt the file.

- **export** - Export the file to the remote server.
- *filename2* - The name you want to assign to the file on the server. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum length of 32 characters.

Note An imported file can contain certificates, RSA or DSA key pairs, or Diffie-Hellman parameters. You must distinguish whether the files contain certificates, privates keys, or Diffie-Hellman parameters by associating the specific contents to a filename. See the "Associating Certificate and Private Key Files with Names" section.

For example, to import the *rsacert.pem* certificate from a remote server to the CSS, enter:

```
# copy ssl sftp ssl_record import rsacert.pem PEM "passwd123"
Connecting
Completed successfully
```

For example, to import the *rsakey.pem* certificate from a remote server to the CSS, enter:

```
# copy ssl sftp ssl_record import rsakey.pem PEM "passwd123"
Connecting
Completed successfully
```

To export the *rsacert.pem* certificate from the CSS to a remote server, enter:

```
# copy ssl sftp ssl_record export rsacert.pem "passwd123"
```

If the **copy ssl** command fails to import certificates or keys, verify the following areas:

- The user account and password in the ftp record are correct
- The base directory is ssh or ssh/path
- The SSH server is reachable
- The SSH server IP address is correct in the ftp-record

Associating Certificate and Private Key Files with Names

After you import or generate certificate and key pair files, you must indicate to the CSS whether these files contain certificates, private keys, or Diffie-Hellman parameters. You do this by associating certificate names, private/public key pair names, or Diffie-Hellman parameter names with the particular imported files.

When you associate the entries specified in the various certificate and private key commands with files, the CSS stores the bindings in the running configuration. Before you log out or reboot the CSS, you must copy the contents of the running-config file to the startup-config file to save the configuration changes and to enable the CSS to use this configuration on subsequent reboots. When you reboot the CSS, the certificate and key associations are loaded automatically.

This section covers:

Associating a Certificate with a File

To associate a certificate name with an imported or generated certificate, use the **ssl associate cert** command. Use the **no** form of the command to remove the association with the file.

The syntax for this command is:

ssl associate cert *certname filename*

The variables are:

- *certname* - The name of the certificate association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the certificate. Enter a maximum of 128 characters. To see a list of imported or generated certificates, use the **ssl associate cert** *certname* **?** command.

For example, to associate the certificate name *myrsacert*1 with the imported certificate file *rsacert.pem*, enter:

```
(config) # ssl associate cert myrsacert1 rsacert.pem
```

To remove the association with the file, enter:

```
(config) # no ssl associate ssl cert myrsacert1
```

Note The **no** form of the command does not function if the associated certificate is in use by an active SSL proxy list.

Associating an RSA Key Pair with a File

To associate an RSA key pair name with an imported or generated RSA key pair, use the **ssl associate rsakey** command. Use the **no** form of the command to remove the association with the file.

The syntax for this command is:

ssl associate rsakey *keyname filename*

The variables are:

- *keyname* - The name of the RSA key pair association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the RSA key pair. Enter a maximum of 128 characters. To see a list of imported or generated RSA keys, use the **ssl associate rsakey** *keyname* **?** command.

For example, to associate the RSA key name *myrsakey1* with the imported *rsakey.pem*, enter:

```
(config) # ssl associate rsakey myrsakey1 rsakey.pem
```

To remove the association with the file, enter:

```
(config) # no ssl associate rsakey myrsakey1
```

Note The **no** form of the command will not function if the associated RSA key pair is in use by an active SSL proxy list.

Associating a DSA Key Pair with a File

To associate a DSA key pair name with an imported or generated DSA key pair, use the **ssl associate dsakey** command. Use the **no** form of the command to remove the association with the file.

The syntax for this command is:

ssl associate dsakey *keyname filename*

The variables are:

- *keyname* - The name of the DSA key pair association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the DSA key pair. Enter a maximum of 128 characters. To see a list of imported or generated DSA keys, use the **ssl associate dsakey** *keyname* **?** command.

For example, to associate the DSA key name *mydsakey1* with the imported *dsakey.pem*, enter:

```
(config) # ssl associate dsakey mydsakey1 dsakey.pem
```

To remove the association with the file, enter:

```
(config) # no ssl associate dsakey mydsakey1
```

Note The **no** form of the command will not function if the associated DSA key pair is in use by an active SSL proxy list.

Associating Diffie-Hellman Parameters with a File

To associate a Diffie-Hellman name with an imported or generated Diffie-Hellman parameter file, use the **ssl associate dhparam** command. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

ssl associate dhparam *paramname filename*

The variables are:

- *paramname* - The name of the Diffie-Hellman parameter association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the Diffie-Hellman parameters. Enter a maximum of 128 characters. To see a list of imported or generated Diffie-Hellman files, use the **ssl associate dhparam** *filename* **?** command.

For example, to associate the Diffie-Hellman filename *mydhparam1* with the imported *dhparams.pem*, enter:

```
(config) # ssl associate dhparam mydhparam1 dhparams.pem
```

To remove the association with the file, enter:

```
(config) # no ssl associate dhparam mydhparam1
```

Note The **no** form of the command will not function if the associated Diffie-Hellman parameter list is in use by an active SSL proxy list.

Verifying a Certificate Against a Key Pair

A digital certificate is built around a public key and can only be used with one key pair. Use the **ssl verify** command to compare the public key in the associated certificate with the public key stored with the associated private key, and verify that they are identical. To see a list of certificate and key pair associations, use the **ssl verify ?** command.

Note If the certificate does not match the public/private key pair, the CSS logs an error message.

The syntax for this command is:

ssl verify *certname keyname*

The variables are:

- *certname* - The association name of the certificate used to verify against the specified key pair.
- *keyname* - The association name of the key pair used to verify against the specified certificate.

For example, to verify the *myrsacert1* digital certificate against the *myrsakey1* key pair, enter:

```
(config)# ssl verify myrsacert1 myrsakey1
Certificate and key match
```

Removing Certificates and Private Keys from the CSS

To remove certificates and private keys from the CSS that are no longer valid, use the **clear ssl file** command. Note that the **clear ssl file** command does not function if the file currently has an association with it. First remove the association to the file by specifying the **no ssl associate** command (see the "Associating Certificate and Private Key Files with Names" section).

The syntax for this global configuration mode command is:

clear ssl file *filename password*

The variables are:

- *filename* - The name of the certificate, key pair, or Diffie-Hellman parameter file that you want to remove from the CSS.
- *password* - The password used to encode the file using DES when it was originally imported or generated by the CSS. This password must be an exact match or the file cannot be cleared.

For example, to remove *dsacert.pem* from the CSS, enter:

```
# clear ssl file dsacert.pem "passwd123"
```